Anpassung von Alignments an Veränderungen in Ontologien

Dissertation

zur Erlangung des Doktorgrades

der Naturwissenschaften

vorgelegt beim Fachbereich 12

der Johann Wolfgang Goethe-Universität

in Frankfurt am Main

von

Matthias Jurisch

aus Georgsmarienhütte

Frankfurt 2021

(D 30)

vom Fachbereich 12 der

Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr.-Ing. Lars Hedrich

Gutachter: Prof. Dr. Uwe Brinkschulte

Prof. Dr. Bodo Igler

Datum der Disputation: 02.07.2021

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Wiesbaden, den 18. Juli 2021

_____

(Matthias Jurisch)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Definitionen und Einschränkungen

Zusammenfassung

Um Wissen in einer Form abzulegen, in der es automatisiert verarbeitet werden kann, werden unter anderem Ontologien verwendet. Ontologien erlauben über einen als Inferenz bezeichneten Prozess die Ableitung neuen Wissens. Bei inhaltlichen Überschneidungen werden Ontologien über Ontologie-Alignments miteinander verbunden, die Entitäten aus den verschiedenen Ontologien in Beziehung zueinander setzen. Üblicherweise werden diese Alignments als Mengen von Äquivalenzen formuliert, die beschreiben, welche Konzepte aus einer Ontologie Konzepten aus einer anderen Ontologie entsprechen. Ebenfalls verbreitet sind Ober- und Unterklassenbeziehungen in Alignments.

Diese Ontologie-Alignments werden zum Beispiel in der Biomedizin in Forschungsdatenbanken verwendet, da durch Alignments Informationen aus verschiedenen Bereichen zusammengeführt werden können. Der manuelle Aufwand, um große Ontologien und Alignments zu erstellen, ist sehr hoch. Dementsprechend wäre es wünschenswert, bei einer Veränderung von Ontologien nicht wieder von vorne beginnen und eine neue Ontologie erstellen zu müssen und möglichst viel aus der veränderten Ontologie und den die Ontologie betreffenden Alignments wiederverwenden zu können. Daher sollten möglichst automatisierte Verfahren verwendet werden. Diese Arbeit untersucht vier Ansätze, um die Anpassung von Alignments an Veränderungen in Ontologien zu automatisieren.

Der erste Ansatz bezieht Inferenzen in den Prozess zur Vorhersage von Alignment-Änderungen mit ein. Dazu werden die Inferenzen vor und nach der Änderung der Ontologien berechnet und auf Basis der Unterschiede mit einem regelbasierten Algorithmus bestimmt, wie sich das Alignment ändern soll. Der zweite Ansatz, wie auch die weiteren Ansätze, hat nicht zum Ziel das Alignment direkt anzupassen. Stattdessen soll vorhergesagt werden, welche Teile des Alignments angepasst werden müssen. Dazu werden die Ontologien und das Alignment als Wissensgraph-Embeddings repräsentiert. Diese Embeddings bilden Knoten aus den Ontologien in einen Raum mit 300-1000 Dimensionen so ab, dass in dem Raum auch die Beziehungen zwischen den Entitäten der Ontologien repräsentiert werden können. Diese Embeddings werden dann verwendet, um verschiedene Klassifikationsalgorithmen zu trainieren. Auf diese Weise wird vorhergesagt, welche Teile des Alignments sich verändern werden. Der dritte Ansatz verbindet Embeddings

mit einem Veränderungsmodell. Das Veränderungsmodell kategorisiert die an den Ontologien vorgenommenen Veränderungen. Auf diese Kategorisierung und das Embedding werden dann Klassifikationsalgorithmen angewandt. Der vierte Ansatz verwendet eine speziell auf Wissensgraphen ausgerichtete Architektur für neuronale Netze, sogenannte Graph Convolutional Networks, um Veränderungen an Alignments vorher zu sagen.

Diese Ansätze werden auf ihre jeweiligen Vor- und Nachteile untersucht. Dazu werden die Verfahren an zwei Anwendungsfällen untersucht. Der Ansatz zur regelbasierten Einbeziehung von Inferenzen wird anhand eines Anwendungsbeispiels aus dem Bereich der Interweaving Systems betrachtet. In dem Beispiel wird eine allgemeine Methode für Interweaving Systems angewandt um das Selbstmanagement von Ampelsteuerungen zu ermöglichen. Die auf maschinellem Lernen aufbauenden Ansätze werden auf einem Auszug aus der biomedizinischen Forschungsdatenbank UMLS evaluiert.

Dabei konnte festgestellt werden, dass die betrachteten Ansätze grundsätzlich zur Anpassung von Alignments an Ontologie-Veränderungen eingesetzt werden können. Der Ansatz zur regelbasierten Einbeziehung von Inferenzen kann dabei vor allem auf sehr kleinen Datensätzen eingesetzt werden, bei denen alle Gesetzmäßigkeiten der Veränderungen grundsätzlich bekannt sind. Diese Anwendbarkeit ergibt sich aus dem Entwurf der Problemstellung für den ersten Ansatz. Die auf maschinellem Lernen aufbauenden Ansätze eignen sich besonders für große Datensätze und bieten den Vorteil, dass auch ohne ein vollständiges Verständnis des Veränderungsprozesses Vorhersagen getroffen werden können. Unter den Ansätzen, die maschinelles Lernen einsetzen, zeigte die Einbeziehung von Veränderungsmodellen keine Vorteile gegenüber den anderen Ansätzen. Auf einem etwas kleineren Datensatz waren die Ergebnisse des Embedding-basierten Ansatzes und der Relational Graph Convolutional Networks vergleichbar, während auf einem größeren Datensatz die Graph Convolutional Networks etwas bessere Ergebnisse erreichen konnten.

Weitere Ergebnisse dieser Arbeit stellen eine Formalisierung der Problemstellung der Anpassung von Ontologie-Alignments an Veränderungen sowie eine formale Darstellung der Ansätze dar. Ein weiterer Beitrag der Arbeit ist die Vorstellung eines Anwendungsfalls aus dem Bereich der Interweaving Systems für Ontologie-Alignments. Außerdem wurde das Problem der Anpassung von Alignments an Veränderungen so formuliert, dass es mithilfe von maschinellem Lernen betrachtet werden kann.

Kapitel 1

Einleitung

Ontologien werden verwendet, um Wissen in einer Form abzulegen, die die automatisierte Verarbeitung dieses Wissens ermöglicht. Ontologien erlauben dabei auch das Ableiten neuen Wissens, das sich aus einer gegebenen Ontologie ergibt. Das Ableiten von neuem Wissen wird als Inferenz bezeichnet. Insbesondere in der Biomedizin und der Abbildung von enzyklopädischen Informationen spielen diese Formen der Wissensrepräsentation eine wichtige Rolle. Wenn sich verschiedene Ontologien inhaltlich überschneiden, werden sie häufig über sogenannte Ontologie-Alignments verbunden. Diese Alignments setzen die Konzepte aus den verschiedenen Ontologien in eine Beziehung zueinander, so dass das Wissen aus beiden Ontologien zusammengeführt werden kann. Typischerweise beschreiben solche Alignments in Form einer Aufzählung von Äquivalenzen, welche Konzepte in einer Ontologie Konzepten aus der anderen Ontologie entsprechen. Ebenfalls verbreitet sind Ober- und Unterklassenbeziehungen in Alignments.

Der Aufwand zum manuellen Erstellen von Ontologien ist sehr hoch. In einer Abschätzung wurde festgestellt, dass in manuell erstellte Ontologien, wie CYC und Freebase, Personalkosten zwischen 2 und 5 Dollar pro Statement anfallen [Pau18]. CYC und Freebase sind Ontologien, die versuchen Konzepte des menschlichen Allgemeinwissens abzubilden und in Beziehung zueinander zu setzen. Da diese Ontologien über sehr viele Statements verfügen führt das zu einem extrem hohen Gesamtaufwand. Automatisiert erzeugte Ontologien wie DBPedia (eine Abbildung von Teilen des Wissens aus Wikipedia in einer Ontologie) erreichen hingegen Kosten im Centbereich pro Statement. Dem entsprechend besteht ein großes Interesse daran, die manuell erstellten Ontologien so lange wie möglich effizient nutzen zu können und wenn möglich Automatisierung einzusetzen. Es existieren keine Untersuchungen zu Kosten von Ontologie-Alignments. Diese werden typischerweise in der Praxis ebenfalls manuell erstellt oder müssen zumindest manuell überprüft werden. Es ist also anzunehmen, dass, um Ressourcen zu sparen, Alignments so weit wie möglich nach Veränderungen wiederverwendet werden sollten und dass

Automatisierung ressourcensparend eingesetzt werden kann.

Zur automatisierten Anpassung von Alignments an die Veränderungen der Ontologien wurden bereits einige Ansätze in der Literatur vorgestellt. Diese Ansätze basieren meist auf Regeln, die für die jeweilige Problemdomäne manuell erstellt werden müssen. Ansätze zum Lernen von solchen Regeln wurden bisher noch nicht entwickelt. Bisher werden für diese Aufgabenstellung einige Aspekte nicht mit einbezogen, die möglicherweise gewinnbringend eingesetzt werden können: Da die Möglichkeit zur Berechnung von Inferenzen eine Kern-Eigenschaft von Ontologien ist, erscheint es sinnvoll, diese direkt in die Betrachtung von Veränderungen mit einzubeziehen. Ein weiterer denkbarer Vorteil der Einbeziehung von Inferenzen währe, dass durch die Einbeziehung von Inferenzen Aspekte wie das Refactoring von Ontologien automatisch mit betrachtet werden können, da die Inferenzen die Semantik abbilden sollen und Refactorings die Semantik nicht verändern sollten. Bisher wurden zur Einbeziehung von Inferenzen noch keine Arbeiten vorgestellt.

Automatisierung durch maschnielles Lernen bei der Anpassung von Alignments an Veränderungen in Ontologien wurde bisher ebenfalls noch nicht betrachtet. Maschinelles Lernen erscheint als sinnvoller Ansatz, da die manuelle Erstellung von Regeln zur Anpassung von Alignments sehr aufwändig ist. Abgesehen davon stellt der Einsatz von maschinellem Lernen eine weitere Möglichkeit der Automatisierung dar und somit eine weitere Möglichkeit zur Einsparung von Ressourcen.

1.1 Forschungsfragestellung und Methodik

Diese Aspekte führen direkt zur Forschungsfrage dieser Arbeit: es soll untersucht werden, in wie weit einerseits die Betrachtung von Inferenzen und andererseits die Verwendung von Methoden des maschinellen Lernens gewinnbringend eingesetzt werden können, um Alignments an Veränderungen in Ontologien anzupassen. Es werden vier Ansätze vorgestellt, um diese Frage untersuchen zu können:

- **Einbeziehung von Inferenzen** Ein Ansatz verbindet regelbasierte Reaktion auf Veränderungen mit der Betrachtung von Inferenzen. Regeln werden dazu auf Unterschiede zwischen den Inferenzen vor und nach der Ontologie-Veränderung angewandt. Die Regeln müssen dabei manuell erstellt werden, da noch keine Systematik zum Lernen von Regeln für diesen Anwendungsfall existiert.

- **Verwendung von Graph-Embeddings** Ein weiterer Ansatz verwendet gelernte Repräsentationen der Struktur von Ontologien, sogenannte Embeddings, in Kombination mit

verbreiteten Klassifikationsverfahren um vorher zu sagen, welche Veränderungen an den Ontologien Veränderungen am Alignment verursachen werden.

- **Kombination von Veränderungsmodell und Embeddings** Der Ansatz, der Emebbdings verwendet, wird um ein Veränderungsmodell erweitert. Das Veränderungsmodell kategorisiert die Art der Veränderungen. Die Embeddings und die Kategorisierungen des Veränderungsmodells werden dann kombiniert. Auf diese Kombination werden dann Klassifikationsverfahren angewandt.

- **Graph Convolutional Networks** Die Embedding-basierten Ansätze arbeiten in zwei Schritten: zuerst werden Embeddings als Repräsentation des Graphen gelernt, dann wird getrennt davon die Klassifikation vorgenommen. Ein weiterer Ansatz untersucht, ob diese Aufgaben auch in einem gemeinsamen Schritt gelernt werden können. Dazu werden Graph Convolutional Networks angewandt. Graph Convolutional Networks sind eine Methode des maschinellen Lernens, mit der Graphdaten direkt verarbeitet werden können.

Diese Ansätze werden anhand verschiedener Anwendungsszenarien evaluiert, da sie unterschiedliche Ziele verfolgen: der regelbasierte Ansatz versucht das Alignment direkt anzupassen, während die auf maschinellem Lernen basierten Ansätze versuchen vorherzusagen, welche Veränderungen an den Ontologien Auswirkungen auf das Alignment haben werden. Für den regelbasierten Ansatz wird ein Anwendungsfall aus dem Bereich der Interweaving Systems vorgestellt, an dem gezeigt wird, wie der Ansatz gewinnbringend eingesetzt werden kann. Der Anwendungsfall betrifft die Selbstorganisation von Ampelsteuerungssystemen, die Daten austauschen und autonom entscheiden sollen, welche Verkehrsdaten ausgetauscht und in den Steuerungsprozess einbezogen werden sollen. Für die auf maschinellem Lernen basierenden Ansätze wird eine Evaluation anhand eines Datensatzes aus der Biomedizin durchgeführt. Der Datensatz enthält Ontologien aus dem Unified Medical Language System (UMLS), das verschiedene biomedizinische Ontologien miteinander verbindet. Für das UMLS liegen Snapshots aus mehreren Jahren vor. Die Evaluation wird so durchgeführt, dass die Ansätze auf einem Versionssprung trainiert und einem anderen anhand von typischen Klassifikationsmetriken bewertet werden.

1.2 Beitrag zum Stand der Forschung

Die Methodik führt zu mehreren Aspekten, die den Stand der Forschung zu Ontologien, Alignments und Veränderungen erweitern. Die wichtigsten Beiträge zum Stand der Forschung sind im folgenden aufgeführt.

- Ein wichtiger Beitrag ist eine Formalisierung der Aufgabenstellung zur Anpassung von Ontologie-Alignments an Veränderungen. In der Arbeit wird eine formale Definition zu Ontologie-Alignments vorgestellt, die diese zusammen mit den Ontologien als Graph repräsentiert. Bereits in dieser Arbeit hat sich gezeigt, dass eine solche Formalisierung hilft, in diesem Umfeld kompaktere Formulierungen anzufertigen.

- Die verschiedenen neuen Ansätze zur Anpassung von Alignments an Veränderungen in Ontologien (der Ansatz zur Einbeziehung von Inferenzen, der Ansatz zur Verwendung von Embeddings, der Ansatz zur Kombination von Embeddings und eines Veränderungsmodells und der Ansatz zur Verwendung von Graph Convolutional Networks) sind der wichtigste Beitrag dieser Arbeit.

- Zur Umsetzung des Ansatzes zur Einbeziehung von Inferenzen wurde ein Framework erstellt. Die Architektur dieses Frameworks ist ebenfalls ein Beitrag dieser Arbeit.

- Die im Rahmen dieser Arbeit angefertigten Implementierungen der Ansätze stellen eine untergeordneten Beitrag dar.

- Ein weiterer Beitrag ist die Vorstellung eines neuen Anwendungsfalls für Ontologie-Alignments im Bereich der Interweaving Systems, der selbstorganisierten Verkehrssteuerungssysteme. Dieser Beitrag ist ein Nebenprodukt der Evaluation des Ansatzes zur Einbeziehung von Inferenzen.

- Zum Export von Veränderungskategorisierungen in Ontologien wurde ein Plugin des Ontologie-Editors Protégé erweitert. Diese Erweiterung wurde den Entwicklern des Plugins als pull-request zur Verfügung gestellt.

- Die neue Formulierung der Anpassung von Ontologie-Alignments an Veränderungen in Ontologien als Problem, das mit maschinellem Lernen bearbeitet werden kann, ist ein weiterer Beitrag dieser Arbeit.

- Die Ergebnisse der Evaluation auf dem biomedizinischen Datensatz stellen ebenfalls einen Beitrag dieser Arbeit zum Stand der Forschung dar. Hier kann beobachtet werden, welche der auf maschinellem Lernen basierenden Ansätze auf dem Datensatz bezüglich verschiedener Metriken welche Leistung erbringen.

- Aus den Ergebnissen können einige Hypothesen über die Ansätze formuliert werden. Auch diese Hypothesen stellen einen Beitrag zum Stand der Forschung dar.

1.3 Struktur der Arbeit

Im folgenden wird der Aufbau dieser Arbeit vorgestellt: Zunächst werden in Kapitel 2 die Grundlagen, die zum Verständnis der Arbeit notwendig sind diskutiert und es wird der Hintergrund der Arbeit vorgestellt. Dazu wird grundlegendes Wissen über Wissensgraphen, Ontologie-Alignments und maschinelles Lernen auf Wissensgraphen präsentiert.

Darauf folgt in Kapitel 3 eine Diskussion der verwandten Arbeiten und der Literatur zu der in dieser Arbeit betrachteten Problemstellung. Zunächst werden andere Arbeiten beschrieben, die Alignments und Veränderungen betrachten. Danach werden entfernter verwandte Problemstellungen, wie die Betrachtung von Veränderungen von Ontologien mit maschinellem Lernen und die Betrachtung von Alignments mit maschinellem Lernen diskutiert. Aus diesen verwandten Arbeiten werden Aspekte extrahiert, die bisher in der Literatur nicht betrachtet wurden, es werden also Forschungslücken identifiziert.

Passend zu den Forschungslücken wird im Anschluss in Kapitel 4 die Problemstellung der Arbeit vorgestellt. Dazu wird zunächst eine grundsätzliche Klassifizierung von Ansätzen zur Anpassung von Ontologie-Alignments diskutiert. Die Klassifizierung betrachtet verschiedene Merkmale der Arbeitsweise dieser Verfahren. Daraufhin wird das Ziel der Arbeit in Form von mehreren Forschungsfragen formuliert. Im Anschluss werden die Aufgabenstellung der Ansätze formalisiert und einige weitere formale Grundlagen, die für die Arbeit wichtig sind, vorgestellt.

Die in dieser Arbeit entwickelten Ansätze werden danach in Kapitel 5 vorgestellt. Zu jedem Ansatz werden außerdem die prinzipiellen Einschränkungen der Anwendung des jeweiligen Ansatzes und die notwendigen Vorbedingungen beschreiben. Zuerst wird der Ansatz zur regelbasierten Einbeziehung von Inferenzen dargestellt. Anschließend werden die auf maschinellem Lernen basierenden Ansätze beschrieben, also die Verwendung von Embeddings, die Kombination von Embeddings und eines Veränderungsmodells und die Verwendung von Graph Convolutional Networks.

Im Anschluss wird in Kapitel 6 die Architektur und Implementierung der Software beschrieben, die erstellt wurde, um die Ansätze einsetzen und evaluieren zu können. Die erstellte Software besteht im wesentlichen aus drei Teilen. Es wurde eine Erweiterung eines Plugins des Ontologie-Editors Protégé erstellt, um Veränderungen in Ontologien erkennen zu können und die Veränderungen in einer maschinenlesbaren Form zu exportieren. Außerdem wurde ein Framework erstellt, mit dem der Ansatz zur regelbasierten Einbeziehung von Inferenzen umgesetzt werden kann. Weiterhin wurde eine Anwendung zur Durchführung von Experimenten mit den Ansätzen, die maschinelles Lernen einsetzen, entwickelt.

Auf die Beschreibung des Entwurfs und der Umsetzung folgt eine Beschreibung der Evaluation der Ansätze in Kapitel 7. Dazu wird zuerst die übergreifende Methodik der Evaluation be-

schrieben und wie die einzelnen Ansätze jeweils evaluiert werden. Die Evaluation selbst besteht vor allem aus zwei Teilen: Zunächst wird anhand eines Anwendungsbeispiel aus dem Bereich des selbstorganisierten Verkehrsmanagements die Anwendung des Ansatzes zur Einbeziehung von Inferenzen demonstriert. Im zweiten Teil werden Ansätze des maschinellen Lernens evaluiert. Dazu wird die Veränderungshistorie eines realen Datensatzes aus der Biomedizin verwendet.

Die Schlussfolgerungen aus der Evaluation werden im Anschluss in Kapitel 8 beschrieben. Zuerst werden die Forschungsfragen beantwortet und in Beziehung zu den vorgestellten Ansätzen gesetzt. Danach werden Vor- und Nachteile sowie verschiedene Eigenschaften der verwendeten Ansätze diskutiert. Ebenfalls vorgestellt werden die Grenzen der Arbeit. Die Diskussion schließt mit einer Betrachtung der allgemeineren Beobachtungen, die sich aus der Arbeit ergeben, ab.

Die Arbeit endet in Kapitel 9 mit einem Fazit und einem Ausblick auf Anknüpfungspunkte für zukünftige Arbeiten.

Kapitel 2

Grundlagen und Hintergrund der Arbeit

Dieses Kapitel stellt die für das Verständnis dieser Arbeit notwendigen Grundlagen vor. Da in dieser Arbeit mit Ontologien gearbeitet wird, wird dazu in Abschnitt 2.1 auf Wissensgraphen und Ontologien eingegangen. Im Zusammenhang mit Wissensgraphen und Ontologien spielen für die vorliegende Arbeit vor allem Ontologie-Alignments eine wichtige Rolle. Dieses Konzept wird in Abschnitt 2.2 vorgestellt. In dieser Arbeit werden unter anderem auch Verfahren des maschinellen Lernens auf Wissensgraphen angewandt. Relevante Verfahren und Grundlagen werden in Abschnitt 2.3 beschrieben.

2.1 Wissensgraphen und Ontologien

Die Kernidee von Wissensgraphen ist es, Wissen in einer strukturierten Form abzulegen, so dass es einfach möglich ist, das Wissen abzufragen und automatisiert zu verarbeiten. Gleichzeitig soll diese Repräsentation so flexibel sein, dass Inhalte, die typischerweise in mehreren verschiedenen Datenbanken abgelegt werden, in einem gemeinsamen Wissensgraph abgelegt werden können. Die übliche Repräsentation wird in Abschnitt 2.1.1 beschrieben. Auf diesen Wissensgraphen können auch formale Mechanismen zur Repräsentierung von konzeptuellem Wissen angewandt werden. Diese formalen Mechanismen werden als Ontologien bezeichnet. Sie erlauben unter anderem auch die automatisierte Ableitung von weiterem Wissen. Ontologien werden in Abschnitt 2.1.2 dargestellt. Bei der Vorstellung von Wissensgraphen und Ontologien wird sich auf die für diese Arbeit relevanten Aspekte konzentriert. Eine umfangreichere aktuelle Übersicht zu Wissensgraphen wird in [Hog+20] vorgestellt.

2.1.1 Wissensgraphen und RDF

Die hier vorgestellte moderne Form von Wissensgraphen wurde insbesondere durch das Semantic Web [BL+01] und den Google Knowledge Graph [Sin12] geprägt. Die Grundidee von Wissensgraphen ist es, Wissen, das normalerweise in textueller Form abgelegt wird, so zu beschreiben, dass es einfacher von Maschinen verarbeitet werden kann. In der Literatur werden Wissensgraphen üblicherweise formal über einen Multigraph mit gelabelten Kanten dargestellt [Hog+20]. Diese Definition wird für die Vorstellung der Grundlagen beibehalten. Im weiteren Verlauf der Arbeit wird eine Definition vorgestellt, die die Notation an den Anwendungsfall anpasst.

Definition 2.1.1 (Wissensgraph) *Ein Wissensgraph $G(V, E, L)$ ist ein Graph mit*

- *V, der Menge der Knoten*
- *L, der Menge der Kantenlabels*
- *$E \subseteq V \times L \times V$, der Menge der Tripel*

Knoten und Kanten-Labels werden jeweils durch Zeichenketten repräsentiert.

Das Semantic Web entstand aus der Idee, das World Wide Web so umzugestalten, dass die im Web abgelegten Informationen so repräsentiert werden, dass sie maschinell verarbeitet werden können und wendet dem entsprechend auch Wissensgraphen an. Zur Umsetzung dieser Idee wird auf zwei zentrale Prinzipien zurückgegriffen:

- Jedem Objekt, das beschrieben wird, wird mindestens ein eindeutiger Internationalized Resource Identifier (IRI) zugewiesen. IRIs sind eine Erweiterung von Uniform Resource Identifiers (URI) auf das Unicode-Character-Set. Dies entspricht den Zeichenketten aus Definition 2.1.1
- Alle Aussagen werden in der Form von Tripeln als `<Subjekt> <Prädikat> <Objekt>` notiert. `Subjekt` und `Prädikat` müssen dabei IRIs sein, `Objekt` kann eine IRI oder ein Literal, also zum Beispiel ein String oder eine Zahl sein. Diese Tripel beschreiben E aus Definition 2.1.1.

Die so geschaffene Struktur ergibt also einen Wissensgraphen entsprechend der Definition 2.1.1. Diese Form ist relativ wenig restriktiv in dem Sinne, dass abgesehen davon, dass sich an die Graph-Form gehalten werden muss, keine weiteren Einschränkungen gelten. Zusätzlich können so abgelegte Daten recht einfach ausgetauscht werden, da ein standartisiertes Format

vorliegt. Die Semantik der so abgelegten Daten wird durch festgelegte Bedeutungen für standartisierte IRIs festgelegt. Ein ähnliches Konzept wurde mit dem Google Knowledge Graph umgesetzt. Im Google Knowledge Graph wird Wissen in einem Graph abgelegt, um Wissen für die automatisierte Verarbeitung, hier zur Unterstützung der Textsuche, verfügbar zu machen. Der Google Knowledge Graph konnte demonstrieren, dass auf Wissensgraphen kommerziell sehr erfolgreiche Anwendungen aufgebaut werden können und etablierte den Begriff der Wissensgraphen.

Das Resource Description Framework (RDF) ist eine Menge von Datenformaten für Wissensgraphen und spezifiziert außerdem ein festgelegtes Vokabular mit definierten Bedeutungen und wurde in [Woo+14] standardisiert. RDF definiert einerseits eine grundsätzliche, von einer textuellen Repräsentation unabhängige Datenstruktur in Form eines Wissensgraphen, eine Menge von IRIs mit standardisierter Bedeutung und eine Menge von textuellen Repräsentationen. Zu diesen Datenformaten gehören ein XML-basiertes Format und mehrere Textformate, die die Tripel-Darstellung von RDF direkt in Text abbilden. Außerdem hat sich um RDF ein breites Feld an Anwendugen, Libraries und Frameworks zum Umgang mit RDF-Daten entwickelt. So wurden APIs wie Apache Jena [Apa20] und die rdflib [rdf20] entwickelt, die jeweils Interfaces für die Arbeit mit RDF-Daten zur Verfügung stellen. Speziell auf die Speicherung von RDF-Daten ausgerichtete Graph-Datenbanken wie zum Beispiel OpenLink Virtuoso [EM09] und stardog [Sta20] wurden ebenfalls entwickelt.

Mit RDF konnten bereits einige große Wissensgraphen umgesetzt werden. Ein sehr populärer Wissensgraph ist DBPedia [Aue+07]. DBPedia hat sich als Ziel gesetzt, Teile des in Wikipedia abgelegten Wissen in RDF-Form verfügbar zu machen. Dazu werden zum Beispiel die Informationen, die in Wikipedia bereits strukturiert abgelegt werden, in ein RDF-Format transformiert. Zu den strukturiert abgelegten Daten gehören zum Beispiel Infoboxen und Artikelkategorien. Ein weiterer großer Wissensgraph ist Wikidata [VK14]. Wikidata verfolgt das Prinzip von Wikipedia, auch bei Wikidata kann der erstellte Wissensgraph kollaborativ bearbeitet werden. Wikidata enthielt im August 2020 Informationen zu über 85 Millionen Entitäten [Wik20b].

Um mit Wissensgraphen zu arbeiten, können Abfragesprachen für Graphen verwendet werden. Diese Abfragesprachen arbeiten üblicherweise so, dass als Anfrage ein sogenanntes Graph-Pattern festgelegt wird, indem es möglich ist, Knoten als Variablen zu definieren. Dann kann in dem abgefragten Graph nach Übereinstimmungen mit dem Pattern gesucht und entsprechende Belegungen der Variablen in dem Graph-Pattern als Ergebnis zurückgegeben werden. Ein Beispiel für eine solche Abfragesprache ist die Abfragesprache SPARQL Protocol and RDF Query Language (SPARQL). SPARQL wurde speziell für die Abfrage auf RDF-Graphen entwickelt.

Die Ausdrucksstärke von SPARQL entspricht der von relationaler Algebra [AG08].

Ein Beispiel für eine SPARQL-Suche über dem Wissensgraphen Wikidata [VK14] ist in Quellcode 2.1 dargestellt. In dem Beispiel kommen eine Knoten-Variable (`?item`) und zwei URIs vor (`wdt:P31` und `wd:Q146`). `wdt:P31` repräsentiert die Beziehung zwischen Instanzen und einer Klasse, `wd:Q146` repräsentiert die Klasse aller Hauskatzen. Dem entsprechend wird nach Einträgen in der Wikidata gesucht, in denen als Prädikat die Instanzbeziehung und im Objekt die Klasse Hauskatze steht. Da diese Statements definieren, dass die Variable `?item` eine Hauskatze ist, wird in diesem Query nach allen in Wikidata repräsentierten Objekten des Typs „Hauskatze“ gesucht. Das Ergebnis ist also eine Liste von Hauskatzen. Bezogen auf die Definition 2.1.1 wird nach Elementen in $< s, p, o > \in E$ gesucht, wobei p die Instanzbeziehung und o die Klasse Hauskatze repräsentiert.

```
1 SELECT ?item WHERE {
2   ?item wdt:P31 wd:Q146.
3 }
```

Listing 2.1: Beispielhaftes SPARQL-Anfrage auf dem Wissensgraphen Wikidata

Die Betrachtung dieser Abfragesprachen zeigen zentrale Vorteile dieser Datenrepräsentation auf: Die Daten können in einem sehr flexiblen Format abgelegt werden. Im Prinzip können alle Arten von Information abgelegt werden. Trotzdem ist es möglich über ein standartisiertes Vokabular eine Art Schema, also eine Formalisierung der Datenstruktur, festzulegen. So kann in RDF zum Beispiel die Zugehörigkeit zu Klassen definiert werden. Ein weiterer Vorteil ist, dass strukturierte Anfragen über diesen Daten durchgeführt werden können, obwohl kein Schema festgelegt werden muss.

Abfragesprachen wie SPARQL werden von den bereits vorgestellten RDF-Datenbanken unterstützt. Damit ergibt sich auch ein zentraler Vorteil aus den großen Wissensgraphen wie Wikidata und DBPedia. So ist es beim Aufbau von enzyklopädischem Wissen nicht mehr notwendig, vorher zu wissen, welche Arten von Anfragen ein Benutzer möglicherweise stellen möchte. Anfragen wie die in Quellcode 2.1 dargestellte Suche nach allen Hauskatzen könnten zum Beispiel in Wikipedia durch Kategorien abgebildet werden, aber kompliziertere Anfragen wie zum Beispiel die Suche nach Personen, denen sowohl ein Oscar als auch der Nobelpreis verliehen wurde, sind nicht möglich. Das letztere Beispiel lässt sich über Wikidata sehr einfach umsetzen, dieses Anfrage wird gemeinsam mit weiteren Beispielen in [Wik20a] dargestellt.

2.1.2 Ontologien

Die gängige Definition von Ontologien wurde in [Stu+98] aufbauend auf der Definition in [Gru+93] vorgestellt: „An ontology is a formal, explicit specification of a shared conceptualisation.“. Daraus ergeben sich drei zentrale Aspekte:

- **conceptualisation**: In der Ontologie wird ein Sachverhalt auf einer abstrakten Ebene beschrieben. Phänomene aus einem Sachgebiet werden in Beziehung zueinander gesetzt und geordnet.
- **formal, explicit specification**: Die Spezifikation wird mithilfe einer formalen Sprache vorgenommen und jedes Konzept wird explizit definiert. Insbesondere soll auch eine formale Semantik definiert sein. Üblicherweise wird die formale Fundierung über eine Form der mathematischen Logik definiert.
- **shared**: Die Ontologie bildet einen Konsens zwischen mehreren Personen oder Systemen ab. Eine Ontologie ist also explizit dazu gedacht, Wissen zu teilen.

Neben dieser Definition, die eher beschreibt, wozu Ontologien verwendet werden können, können Ontologien auch formal betrachtet werden. Dazu werden Ontologien als Wissensbasen auf der Grundlage von Beschreibungslogiken definiert. In dieser Arbeit werden Ontologien vor allem als Wissensgraphen betrachtet, die über eine formale Semantik verfügen und aus denen neue Aussagen abgeleitet werden können. Wie dieser Inferenzmechanismus funktioniert und wie die konkrete Semantik genau aussieht, spielt dabei keine Rolle. Um trotzdem einen Eindruck der typischen Formulierung von Semantiken im Kontext von Ontologien zu erhalten, wird im folgenden beispielhaft die Definition einer Wissensbasis in der Beschreibungslogik $\mathcal{AL}$ betrachtet, da diese relativ kompakt und verständlich definiert werden kann und in den in der Praxis verwendeten Ontologiesprachen enthalten ist. Eine ausdrucksstärkere Beschreibungslogik wie sie für die meisten Ontologiesprachen verwendet wird, wird hier nicht vorgestellt, da in dieser Arbeit Beschreibungslogiken nur am Rande betrachtet werden. Zunächst muss dazu die Syntax der $\mathcal{AL}$ dargestellt werden:

Definition 2.1.2 (Konzeptbeschreibungen in $\mathcal{AL}$[Baa+10]) *Für Konzeptbeschreibungen in $\mathcal{AL}$ gilt die folgende syntaktische Erstezungsregel für die Symbole C und D:*

$C, D \rightarrow$

- *A: atomares Konzept*
- *$\top$: universelles Konzept*

- $\bot$*: leeres Konzept*
- $\neg A$*: atomare Negation*
- $C \sqcap D$*: Konzept-Schnitt*
- $\forall R.C$*: Wert-Restriktion*
- $\exists R.\top$*: existenzielle Quantifizierung*

Dabei ist ein atomares Konzept die Beschreibung einer Menge, $\top$ steht für die Vereinigungsmenge aller Mengen, $\bot$ steht für die leere Menge und $\neg A$ ist die Menge aller Objekte, die nicht in der durch das atomaren Konuzept A beschriebenen Menge enthalten sind. Der Konzept-Schnitt repräsentiert die Menge aller Objekte, die in den geschnittenen Klassen enthalten sind. Die Wert-Restriktion beschreibt die Menge der Individuen, die über eine Relation R mit Elementen aus einer bestimmten Menge C verbunden sind und die existenzielle Quantifizierung beschreibt die Menge der Individuen, die über die Relation R mit mindestens einem anderen Individuum verbunden sind. Mit dieser Syntax kann zum Beispiel die folgende Formel dargestellt werden:

$$\exists hasChild.\top \sqcap Male$$

Wenn die Menge $Male$ die Menge aller männlichen Personen und $hasChild$ die Beziehung zwischen Eltern und deren Kindern beschreibt, dann repräsentiert die dargestellte Formel die Menge aller Väter.

Über die Beschreibung von Konzepten hinaus können in $\mathcal{AL}$ terminologische Axiome formuliert werden:

Definition 2.1.3 (Terminologische Axiome) *Terminologische Axiome in $\mathcal{AL}$ mit den Konzeptbeschreibungen C und D sind die folgenden Axiome:*

- $C \sqsubseteq D$
- $C \equiv D$

Terminologische Axiome werden genutzt, um Beziehungen zwischen Konzeptdefinitionen herzustellen. Dabei repräsentiert $\sqsubseteq$, dass ein Konzept in einem anderen enthalten ist und $\equiv$, dass zwei Konzepte äquivalent sind.

Weiterhin kann in der $\mathcal{AL}$ Individuen ein Konzept zugewiesen werden. Diese Zuweisungen werden als assertionale Axiome bezeichnet:

Definition 2.1.4 (Assertionale Axiome) *Sei C ein Konzept, R eine Relation und a, b und c Individuen dann sind die Folgenden Formeln assertionale Axiome:*

- $C(a)$
- $R(b, c)$

Diese assertionalen Formulierungen werden genutzt um Individuen Klassen zuzuweisen und Individuen in Beziehung zueinander zu setzen.

Mit diesen formalen Mitteln kann eine Wissensbasis definiert werden. Ontologien, die in Beschreibungslogik formuliert werden, sind Wissensbasen. Eine Wissensbasis wird basierend auf der Beschreibungslogik wie folgt definiert [Baa+10]:

Definition 2.1.5 (Wissensbasis) *Eine Wissensbasis $\mathcal{K}(\mathcal{T}, \mathcal{A})$ wird in eine T-Box $\mathcal{T}$ und eine A-Box $\mathcal{A}$ aufgeteilt. Dabei gilt:*

- *$\mathcal{T}$ enthält terminologische Axiome.*
- *$\mathcal{A}$ enthält assertionale Axiome.*

Für die technische Umsetzung von Ontologien wird häufig die Web Ontology Language (OWL) [MVH+04] eingesetzt. OWL ist, wie auch RDF, aus den Ideen des Semantic Web entstanden. OWL erlaubt es formale Definitionen zu RDF-Dokumenten hinzuzufügen. Dazu existiert eine Menge an RDF-URIs mit entsprechender Semantik. Die Semantik dieser Definitionen wird in der neuesten Version OWL-2 [Hit+09] mit der Beschreibungslogik $\mathcal{SROIQ}$ [Hor+06] umgesetzt, die $\mathcal{AL}$ enthält. Das $\mathcal{S}$ in $\mathcal{SROIQ}$ steht als Abkürzung für $\mathcal{ALC}$, eine Erweiterung der $\mathcal{AL}$, die auch die Negation von nicht-atomaren Konzepten erlaubt.

Die Beschreibungslogiken, die für Ontologien verwendet werden sind auf Teilmengen der Prädikatenlogik abbildbar. Dem entsprechend ist es unter bestimmten Bedingungen möglich, weitere Fakten zu solchen Ontologien abzuleiten und Aussagen über die Konsistenz von Wissensbasen zu treffen. Die Betrachtung der Semantik kann mithilfe von Software unterstützt werden. Diese Software wird üblicherweise als *Reasoner* bezeichnet. Ein Reasoner erlaubt es zum Beispiel zusätzliche Fakten aus einer Ontologie abzuleiten oder zu überprüfen, ob eine Ontologie konsistent ist. Reasoning auf OWL-Ontologien ist sehr aufwändig, auf Teilsprachen der OWL 1 ist Reasoning teilweise unentscheidbar. Die Weiterentwicklung von OWL 1, OWL 2, verfügt deswegen über spezialisierte Teilsprachen, die bestimmte Aufgaben, unter anderem das effizientere Reasoning, erleichtern sollen. Auf OWL 2-Ontologien liegt für die einfachsten Teilsprachen das Reasoning in der Klasse NL, also bei logarithmischem Speicherplatzbedarf auf nichtdeterministischen Turingmaschinen [Mot+12]. Es existieren zahlreiche Reasoner für

OWL, im bereits erwähnten Virtuoso ist zum Beispiel ein Reasoner integriert. Andere populäre Reasoner für OWL sind Fact++ [TH06] und Pellet [Sir+07].

Zur Bearbeitung von Ontologien hat sich Protégé [Noy+01] als populärste Ontologie-Entwicklungsumgebung etabliert. Protégé ist ähnlich wie Eclipse ein erweiterbares Framework, zu dem viele Plugins existieren und wird als freie Software veröffentlicht, was unter anderem das Einbinden von verschiedenen Reasonern und Visualisierungswerkzeugen ermöglicht. Es existiert unter anderem auch eine Erweiterung zum Erkennen von Veränderungen in Ontologien, owl-diff [Kre+11].

Auch für Ontologien gibt es einige praktische Anwendungsbeispiele. Ein sehr frühes Anwendungsfeld für Ontologien waren Wissensdatenbanken in der Biomedizin. Bereits 1986 wurde ein sogenannter Metathesaurus mit maschinenlesbaren Informationen zum Zusammenführen verschiedener Terminologischer Systeme aus der Medizin vorgestellt [Lin+93]. Auch diese Terminologien werden heute in Form von Ontologien dargestellt. In der Gene Ontology [Con04] werden als ein weiteres Beispiel aus der Biomedizin Informationen zu Genen abgelegt. Im Bereich des E-Commerce wird ebenfalls mit Ontologien gearbeitet: Die Ontologie Goodrelations [Hep08] stellt eine Ontologie zur Beschreibung von Produkten vor. Das in Goodrelations verwendete Vokabular kann zum Beispiel verwendet werden, um Produkte auf Webseiten zu annotieren. Suchmaschinen wie Google Shopping verwenden diese Annotationen, um so annotierte Produkte in der Suche mit entsprechenden Informationen anzuzeigen.

2.2 Ontologie-Alignments

Die hier vorgestellte Arbeit betrachtet nicht nur einzelne Ontologien, sondern auch sich überlappende Ontologien, die durch sogenannte Ontologie-Alignments verbunden werden. Um hier den entsprechenden Hintergrund bereit zu stellen, werden im Folgenden Ontologie-Alignments vorgestellt, und es wird beschrieben, wie Alignments erzeugt werden können. Dazu wird sowohl vorgestellt, wie manuell Alignments erzeugt werden (Abschnitt 2.2.1) als auch wie diese automatisiert generiert werden können (Abschnitt 2.2.2).

Als *Ontology Alignment* wird eine Menge von Verbindungen zwischen zwei Ontologien betrachtet. Das Alignment ist das entscheidende Format für die Verbindung von Ontologien, da es den Zusammenhang zwischen den Ontologien festlegt. Die einzelnen Verbindungen, also die Elemente des Alignments, werden als Korrespondenzen bezeichnet. [ES07] definiert Korrespondenzen wie folgt:

Definition 2.2.1 (Korrespondenz) *Für Zwei Ontologien O und O' mit der Menge an atomaren Konzepten Q_O und $Q_{O'}$ und einer Menge von Alignment-Relationen Θ ist eine Korre-*

spondenz ein Tripel

$$\langle e, e', r \rangle$$

mit

- $e \in Q_L$ *und* $e' \in Q_{L'}$
- $r \in \{\sqsubseteq, \equiv\}$

Damit definiert [ES07] ein Alignment auf folgende Art:

Definition 2.2.2 (Alignment) *Seien o und o' Ontologien, dann ist ein Alignment eine Menge von Korrespondenzen zwischen Paaren von Entitäten aus $Q_L(o)$ und $Q'_L(o')$.*

Diese Definition wird in dieser Arbeit nur für die Vorstellung von Ontologie-Alignments verwendet. Im weiteren Verlauf der Arbeit werden weitere, auf den jeweiligen Anwendungsfall angepasste Definitionen von Alignments vorgestellt, die die Ontologien und Alignments in einer gemeinsamen Definition betrachten. Eine wichtige Eigenschaft von Alignments ist die Fragestellung, auf wie viele Entitäten aus der jeweils anderen Ontologie eine Entität abgebildet werden kann. Kommt jede Entität in einem Alignment nur ein mal vor, spricht man von einem 1:1 Matching. Ansonsten wird das Matching als ein n:m-Matching bezeichnet.

2.2.1 Manuelles Alignment

Alignments können manuell erstellt werden. Der übliche Prozess für das Alignment von Datenquellen wird in [Stu10] beschrieben. Dabei wird eine zusätzliche, als „neutrale Ontologie“ bezeichnete Ontologie definiert, die die Überschneidungen der Ontologien abbildet. In dieser neutralen Ontologie wird also ein übergeordnetes Gerüst erstellt, dass alle semantischen Informationen enthält, die in den in Beziehung stehenden Ontologien vorhanden sind. Dann werden die Begriffe aus diesen Ontologien in ein Verhältnis zu den übergeordneten Ontologien gesetzt. Daraus lässt sich dann ein Alignment extrahieren. Wie sich bereits aus dieser kurzen Beschreibung ersehen lässt, ist dieser Prozess sehr aufwändig und benötigt viel manuelle Arbeit.

2.2.2 Ontology Matching

Ontology Matching bezeichnet den Prozess des automatisierten Erstellen eines *Ontology Alignments*. Als Eingabe dienen zwei Ontologien sowie Parameter und möglicherweise externes

Wissen, wie zum Beispiel ein Thesaurus oder ähnlich beim Mapping-Prozess nützliche Informationen. Ausgabe ist ein wie in Definition 2.2.2 beschriebenes Alignment. Zur automatisierten Erstellung von Ontologie Alignments hat sich eine aktive Community entwickelt. Dazu werden seit 2004 jährliche Workshops durch die Ontology Alignment Evaluation Initiative (OAEI) durchgeführt, bei denen verschiedene Werkzeuge auf Basis von vorgegebenen Datensätzen miteinander verglichen werden [Jr20].

Ein Survey zu Ontology Matching findet sich in [SE13]. Die dabei angewendeten Methoden werden in die Kategorien der terminologischen, strukturellen, extensionalen und semantischen Vorgehen eingeordnet. Terminologische Ansätze verwenden Ähnlichkeiten in den Namen von Entitäten, um Korrespondenzen zu finden. Die strukturellen Ansätze arbeiten auf der Ähnlichkeit der dargestellten Graphen. Extensionale Methoden versuchen Ähnlichkeiten auf der Instanz-Ebene zu finden, um darauf basierend Korrespondenzen zu generieren. Semantische Ansätze arbeiten mit möglichen Interpretationen der Ontologien. In dem Survey werden *SAMBO* [LT06], *Falcon* [Hu+08], *DSSIM* [NVV11], *ASMOV* [JM+09], *RiMOM* [Li+09], *AnchorFlood* [SA09] und *AgreementMaker* [Cru09] kurz vorgestellt und nach Vorgaben der Ontology Alignment Evaluation Initative verglichen. Diese Tools verwenden alle mehrere Ansätze, wobei alle auf terminologische und strukturelle Verfahren zurückgreifen. Extensionale Ansätze werden von vier der drei untersuchten Tools verwendet, Semantische Ansätze nur von zwei Tools. Eine weitere wichtige Feststellung dieses Vergleichs ist, dass alle Tools in der Lage sind, mit OWL zu arbeiten und einige auch RDFS und SKOS anbieten.

Eine breitere Literaturübersicht bietet [OC+15]. In dieser Meta-Analyse werden die Veröffentlichungen zu Ontology Matching der vorherigen 10 Jahre verglichen und kurz analysiert, wie sich die Themenstellung verändert hat, sowie die in Veröffentlichungen präsentierten Tools kurz vorgestellt. Darauf aufbauend werden Forscher aus dem Gebiet zu ihren Vorstellungen der Veränderungen in diesem Themengebiet befragt. Ein wichtiges Ergebnis dieser Befragung ist, dass für eine zentrale Forschungsfrage noch keine Antwort gefunden wurde. Diese Frage ist, wie es möglich ist, das Finden von nicht nur 1:1 sondern auch n:m-Matchings besser zu unterstützen. Die Forscher gaben an, dass diese Fragestellung insbesondere von Bedeutung ist, da in der Praxis bei großen Ontologien häufig von einem n:m-Matching ausgegangen werden muss.

2.2.3 Beispiele für Alignment-Ansätze

In diesem Abschnitt wird jeweils ein Beispiel für einen Ansatz aus den terminologischen (Abschnitt 2.2.3), strukturellen (Abschnitt 2.2.3), extensionalen (Abschnitt 2.2.3) und semantischen Vorgehen (Abschnitt 2.2.3) beispielhaft vorgestellt. Dies soll dazu führen, einen konkreteren

Einblick in die tatsächlichen Methoden zu bekommen, und damit auch einen Überblick über das Feld zu erhalten. In dieser Arbeit werden solche Methoden nicht verwendet, da die Methoden auf den betrachteten Anwendungsfällen nicht anwendbar sind. Ein Matching-Tool verwendet allerdings normalerweise mehrere verschiedene Matching-Algorithmen und kombiniert diese entsprechend.

Cosynonym-Ähnlichkeit

Das Finden von Alignments basierend auf Cosynonym-Ähnlichkeiten [ES07] ist ein Beispiel für eine terminologische Methode. Dieser Ansatz verwendet als externe Quelle sogenannte „synonym resources“, eine spezielle Form von Thesauri.

Definition 2.2.3 (Synonym Resource) *Eine Synonym Resource Σ über einer Menge von Wörtern W ist ein Tupel $\langle E, \lambda \rangle$ mit der Menge von Synonym-Mengen $E \subseteq 2^W$ und der Funktion λ, die jeder Synonym-Menge eine Beschreibung zuzuweist. Für einen Term t wird die Anzahl der assozierten Synonym-Mengen mit $\Sigma(t)$ notiert.*

Eine Synonym-Ressource erlaubt es also, einem Term Synonyme zuzuordnen. Damit kann die Cosynonym-Ähnlichkeit definiert werden:

Definition 2.2.4 (Cosynonym-Ähnlichkeit) *Seien s und t Terme und Σ eine Synonym Resource, dann ist die Cosynonym-Ähnlichkeit σ definiert als*

$$\sigma(s,t) = \frac{|\Sigma(s) \cap \Sigma(t)|}{|\Sigma(s) \cup \Sigma(t)|}$$

Die Cosynonym-Ähnlichkeit vergleicht die Mengen der Synonyme, um einen Aufschluss darüber zu geben, wie ähnlich die Bedeutung der Begriffe ist. Vergleicht man Konzepte aus zwei Ontologien, lässt sich dieses Vorgehen auf die Label der verschiedenen Konzepte anwenden, um Korrespondenzen zu finden.

Structural Topological Dissimilarity

Structural Topological Dissimilarity [VE97] ist eine einfache Metrik, um in einer Taxonomie die Ähnlichkeit zwischen Konzepten zu finden. Um diesen Ansatz auf Ontology Matching anwenden zu können, müssen zwischen den Ontologien bereits Beziehungen hergestellt worden sein. Die Metrik selbst ist in [ES07] wie folgt definiert:

Definition 2.2.5 (Structural topological dissimilarity) *Die Structural topological dissimilarity $\delta : o \times o \to \mathbb{R}$ ist eine dissimilarity über der Hierarchie $H = \langle o, \leq \rangle$, so dass*

- $\forall e, e' \in o, \delta(e, e') = \min_{c \in o}[\delta(e, c) + \delta(e', c)]$
- $\delta(e, c)$ *ist die Anzahl der Kanten zwischen* e *und einem anderen Element* c*.*

In einem Beispiel für Ontologien könnte man dieses Matching auf der durch `rdfs:subClassOf` entstehenden Klassenhierarchie durchführen. Da bereits gemeinsame Knoten bestehen, können diese als Referenzknoten c für die Berechnung der dissimilarity genutzt werden.

Formale Begriffsanalyse

Ein Beispiel für eine extensionale Methode ist *Formale Begriffsanalyse* [GW97]. Extensionale Methoden versuchen aus einer gemeinsamen Basis von Instanzen, die in beiden Ontologien vorkommen, gemeinsame Klassen in den Ontologien zu finden. Dafür müssen die Instanzen in beiden Ontologien allerdings bekannt sein (und die Äquivalenz der Instanzen in den verschiedenen Ontologien). Formale Begriffsanalyse geht davon aus, dass dies bereits der Fall ist. Es gibt Verfahren, um Äquivalenzen zwischen Instanzen in verschiedenen Ontologien zu finden, diese werden hier allerdings nicht betrachte.

Die formale Begriffsanalyse erzeugt ein Konzeptgitter aus einer Menge von Objekten und deren Eigenschaften. Bei dem Anwendungsfall „Ontology Matching“ werden die Instanzen als Objekte betrachtet und die Zugehörigkeit zu den Klassen der beiden Ontologien als Eigenschaften. Begonnen wird dabei mit der Potenzmenge der Menge an Klassen. Für jedes Element aus dieser Menge wird daraufhin untersucht, ob es eine eindeutige Menge an Eigenschaften gibt, die dem Element zugeordnet werden können, ansonsten wird es als Wurzel betrachtet. Wenn die Wurzel das Element bereits enthält, wird es entfernt.

Ein Beispiel für die Eingabe einer möglichen Anwendung dieses Verfahrens ist in Tabelle 2.1 dargestellt. Die Instanzen sind sowohl in den zu mappenden Ontologien 1 und 2 bekannt. Die Zugehörigkeit zu einer Ontologie ist für jedes Konzept mit dem Index angegeben. Die Zugehörigkeit der Instanzen zu den Klassen ist jeweils mit einem Haken dargestellt.

Instanzen/Klassen	$Fahrzeug_1$	$Flugzeug_1$	$Boot_2$	$Flugzeug_2$
A380	√	√		√
Boeing 747	√	√		√
Ruderboot	√		√	
VW Käfer	√			

Tabelle 2.1: Beispiel für die Instanzen als Eingabe für Formale Konzeptanalyse

Das Ergebnis der Formalen Begriffsanalyse ist in Abbildung 2.1 dargestellt. Das Konzept $Fahrzeug_1$ ist als Wurzel des Baumes berechnet worden. Die Klassen $Flugzeug_1$ und $Flugzeug_2$

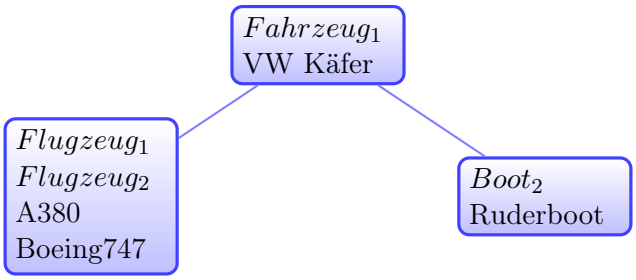

Abbildung 2.1: Ergebnis der Formalen Begriffsanalyse

sind als eine Unterklasse und $Boot_2$ als eine andere Unterklasse von $Fahrzeug_1$ bestimmt worden. Aus dem Konzeptgitter lassen sich nun die folgenden Klassenbeziehungen ableiten:

$$Flugzeug_1 \equiv Flugzeug_2$$

$$Flugzeug_1 \sqsubseteq Fahrzeug_1$$

$$Flugzeug_2 \sqsubseteq Fahrzeug_1$$

$$Boot_2 \sqsubseteq Fahrzeug_1$$

Verwendung von SAT-Solvern

In [GS04] wird ein Ansatz vorgestellt, der Semantisches Matching auf der Basis von SAT-Solvern vornimmt. Auch bei diesem Vorgehen wird ein initiales Matching benötigt, das als Basis für ein verfeinertes Matching dient. Dabei werden die Klassenhierarchie der Ontologien und das Matching in aussagenlogische Formeln umgewandelt.

Für alle Paare von Klassen aus Ontologien kann dann ein Axiom für die denkbaren Beziehungen $(=, \leq, \geq, \perp)$ zwischen den Klassen bestimmt werden. Mit dem SAT-Solver kann daraufhin untersucht werden, welche von diesen Axiomen direkt aus dem bereits angegebenen Matching folgen. Dieser Vorgang kann wiederholt werden, bis keine neuen Klassenbeziehungen mehr abgeleitet werden.

2.3 Maschinelles Lernen auf Wissensgraphen

In diesem Abschnitt werden verschiedene Aspekte aus dem Bereich des maschinellen Lernens auf Wissensgraphen vorgestellt. Da unter anderem mit Embeddings von Wissensgraphen gearbeitet wird, werden diese Verfahren in Abschnitt 2.3.1 vorgestellt. In Kombination mit diesen Embeddings werden Klassifikationsverfahren verwendet. Eine Übersicht über relevante Verfahren und

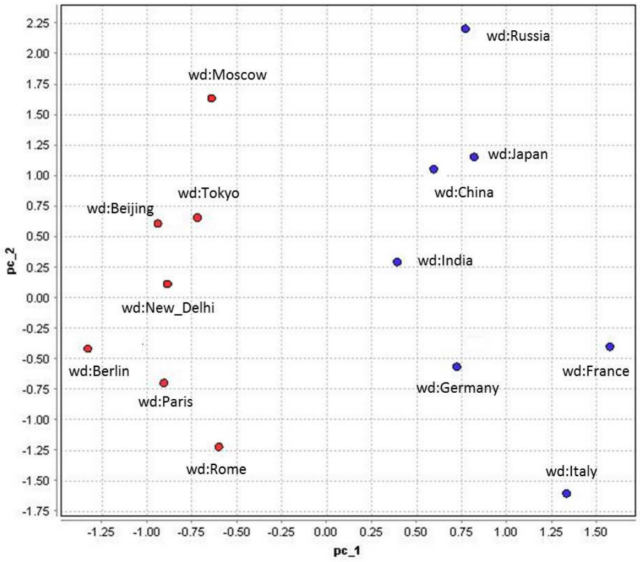

Abbildung 2.2: Hauptkomponentenanalyse eines Ausschnitts aus einem Embedding-Raum für Wikidata [RP16]

Metriken, die zur Evaluation dieser Verfahren genutzt werden können, wird in Abschnitt 2.3.2 dargestellt.

2.3.1 Embeddings von Wissensgraphen

Graph-Embeddings werden in dieser Arbeit verwendet, um Graphen zu repräsentieren. Embeddings sind Abbildungen von Objekten (zum Beispiel Worte oder Graphknoten) in einen Vektorraum mit typischerweise 300-1000 Dimensionen. Üblicherweise wird jedem Objekt dabei ein Vektor in diesem Vektorraum zugeordnet:

Definition 2.3.1 (Embedding) *Ein Embedding τ einer Menge von Objekten O ist eine Abbildung von Graph-Knoten in den $\mathbb{X}^n$:*

$$\tau : O \rightarrow \mathbb{X}^n$$

n kann dabei frei gewählt werden.

In den meisten Fällen gilt $X = \mathbb{R}$ abgebildet.

Die Abbildung der Objekte in den Vektorraum X^n wird dabei gelernt. In Abbildung 2.2 wird eine zweidimensionale Hauptkomponentenanalyse des Embeddings von Hauptstädten und Ländern aus Wikidata dargestellt. Eine Hauptkomponentenanalyse ist eine Methode, um Daten aus einem Raum mit vielen Dimensionen in einem Raum mit weniger Dimensionen darzustellen, so dass sich Muster in den Daten erkennen lassen [Jac05]. Das zugrundeliegende Embedding

verfügt über 500 Dimensionen. In der Abbildung lässt sich beobachten, dass die Beziehung zwischen Hauptstädten und dem jeweiligen Staat sich als Translation im Vektorraum manifestiert. Um nachvollziehen zu können, wie diese Embeddings erzeugt werden und welche Art von Informationen mit den in dieser Arbeit verwendeten Graph-Embeddings dargestellt werden können, werden im Folgenden einige Verfahren zum Embedding von Wissensgraphen kurz vorgestellt. Diese Verfahren werden in dieser Arbeit in drei Kategorien unterteilt: (1) Verfahren, die an Verfahren aus dem Natural Language Processing angelehnt sind, (2) Knowledge-Base-Completion-Verfahren, die auf Translationen basieren und (3) Knowledge-Base-Completion-Verfahren, die auf Faktorisierung basieren.

(1) **Sprachmodellierung** Im Bereich des Natural Language Processing wurden Verfahren wie Word2Vec [Mik+13] und Glove [Pen+14] vorgestellt. Ziel dieser Verfahren ist es, unüberwacht eine Vektor-Repräsentation für Worte aus großen Textcorpora zu lernen. Dazu wird der Kontext, in dem Worte in dem Textcorpus verwendet werden, betrachtet. Word2Vec versucht mithilfe neuronaler Netzwerke Vorhersagen über solche Kontexte zu treffen. Aus den Gewichten im neuronalen Netzwerk lassen sich dann die Embeddings bilden. Glove bildet Matritzen, die abbilden, wie oft Worte im Kontext anderer Worte auftauchen. Diese sogenannte Kookkurenzmatrix wird dann faktorisiert, so dass sich für jedes Wort ein Vektor ergibt.

Solche Ansätze aus der Sprachmodellierung können auch auf Wissensgraphen angewandt werden. Dazu muss eine Strategie gefunden werden, wie Beispiele für Kontexte von Graphknoten erzeugt werden können. Im folgenden wird diese Strategie für RDF2Vec näher erläutert, da dieses Verfahren in dieser Arbeit verwendet wird. RDF2Vec [RP16] erzeugt Beispiele für Kontexte, indem zufällige Pfade durch den Graphen erzeugt werden. Diese Pfade werden als Random Walks bezeichnet. Da sowohl beliebig viele als auch beliebig lange Pfade generiert werden könnten, werden nur wenige Pfade zufällig ausgewählt. Diese Pfade werden dann als Eingaben für Word2Vec verwendet. Betrachtet man die Modelle genauer, wird formal versucht für zwei Verfahren Vektoren so zu wählen, dass die Scoring-Funktion für die im Corpus enthaltenen Beispiele möglichst groß wird.

Die Scoring-Funktion für das erste Verfahren, das als Continuous Bag-of-Words-Modell bezeichnet wird, repräsentiert, wie wahrscheinlich es ist, dass ein Wort in einem gegebenen Kontext vorkommt. Die Scoring-Funktion lautet wie folgt:

$$p(w_t|w_{t-c}...w_{t+c}) = \frac{e^{(v^T v'_{w_t})}}{\Sigma_{w=1}^{V} e^{(\bar{v}^T v'_w)}} \tag{2.1}$$

dabei ist w_t das Wort für das der Score berechnet wird und w_{t-c} und w_{t+c} der vorgegebene Kontext. Der Index eines Worts repräsentiert die Position des Worts in einem Satz. $\bar{v}$ ist der

Durchsnittsvektor der Wörter, die den Kontext repräsentieren, v'_w ist der Output-Vektor eines Worts und V ist das Vokabular, also die Menge aller Wörter.

Das zweite Verfahren wird als Skip-Gram-Modell bezeichnet. Diese Scoring-Funktion soll die Wahrscheinlichkeit codieren, dass ein Wort im Kontext eines anderen Wortes steht. Die Scoring-Funktion für das Skip-Gram-Modell lautet wie folgt:

$$p(w_o|w_i) = \frac{e^{(v'_{w_o} v_{w_i})}}{\Sigma_{w=1}^{V} e^{(v'^T v_{w_i})}} \tag{2.2}$$

dabei sind w_i und w_o Worte. Der Score soll bestimmen, mit welcher Wahrscheinlichkeit w_0 im Kontext von w_i steht. v'_w ist der Ausgabevektor eines Worts und v_w der Eingabevektor eines Worts. Der Eingabevektor ist dabei eine interne Repräsentation, die zwar gelernt aber nicht weiter verwendet wird. V repräsentiert das Vokabular. Diese Funktion wird für alle Worte w_o im Kontext von w_i berechnet.

Als Kontext werden dann die Random Walks betrachtet und als Worte jeweils die URIs der Graphknoten und die URIs der Relationen in den Random Walks. So kann ein Embedding für jeden Graphknoten und für alle Relationen generiert werden.

(2) **Translative Modelle** Weitere Modelle stammen aus dem Bereich der Knowledge Base Completion. Diese Verfahren nutzen Vektorrepräsentationen, um zusätzliche Verbindungen in Wissensgraphen zu finden. Translative Modelle versuchen, die Embeddings so zu wählen, das alle Knoten, die durch eine bestimmte Relation verbunden werden, durch die gleiche Translation in einem Vektorraum verbunden werden können. Das erste solche Verfahren wurde als TransE vorgestellt [Bor+13]. In TransE werden alle Entitäten als Vektoren und alle Relationen als ein Verschiebungsvektor repräsentiert. Auch hier wird eine Scoring-Funktion für im Trainingsdatensatz vorhandene Beispiele maximiert. Beispiele werden üblicherweise aus dem Wissensgraph generiert. Da auch negative Beispiele benötigt werden, müssen diese ebenfalls aus dem Graphen generiert werden. Die Scoring-Funktion für ein Tripel bei gegebenem Wissensgraph $G(V, E, L)$ ist:

$$p(h, l, t) = ||v_h + v_l - v_t||_n, < h, l, t > \in E \tag{2.3}$$

Dabei bildet $||x||_l$ die l-Norm eines Vektors ab und v_h, v_r und v_t sind jeweils die Vektorrepräsentationen von h, r und t. Aus dieser Scoring-Funktion ergibt sich, dass es nicht möglich ist, $1 : n$ und $n : m$ Relationen adäquat zu repräsentieren. Wird zum Beispiel die Relation $\leq$ der natürlichen Zahlen betrachtet, müssten die Zahlen 1, 2 und 3, wie auch alle anderen natür-

lichen Zahlen an einen Ort im Vektorraum abgebildet werden, da alle Zahlen in dieser Relation zur Zahl 1 stehen. Das ist offensichtlich kein wünschenswertes Verhalten für ein Embedding-Verfahren.

TransH [Wan+14] erweitert die grundsätzliche Idee von TransE um die Möglichkeit, $n:m$-Relationen abzubilden. Die Idee dieses Ansatzes ist es, für jede Relation eine Hyperebene zu lernen und die Repräsentationen der Graphknoten darauf zu projizieren. Relationen sollen dann durch Translationen auf der zur Relation gehörenden Hyperebene repräsentiert werden. Dazu wird ein Normalenvektor w für die Relationsebene verwendet, um einen Vektor auf die Ebene zu projizieren:

$$v^r = v - wr^T vw \tag{2.4}$$

Diese Projektion wird dann zur Berechnung der Scoring-Funktion genutzt:

$$p(h,l,t) = ||v_h^r + v_l^r - v_t^r||_2^2, < h,l,t > \in E \tag{2.5}$$

So lassen sich auch $n:m$-Relationen abbilden: die Graphknoten, die in einer n:m-Beziehung stehen, müssen nicht mehr an einen Ort im Vektorraum abgebildet werden, da durch die Projektion auf die Relationsebene unterschiedliche Vektoren auf den gleichen Vektor auf der Relationsebene abgebildet werden können. Es existieren noch einige weitere translative Verfahren zur Erstellung von Knowledge-Graph-Embeddings, wie zum Beispiel TransD [Ji+15] oder TransR [Lin+15]. Diese Verfahren werden an dieser Stelle allerdings nicht weiter betrachtet, da sie in dieser Arbeit nicht verwendet werden, da unklar ist, ob diese Verfahren tatsächlich einen Vorteil bringen können. In [Kad+17] konnte gezeigt werden, dass einfachere Varianten bei entsprechender Anpassung der Hyperparameter besser als komplexe Erweiterungen abschneiden.

③ **Faktorisierung**: Ein weiterer Ansatz zur Repräsentierung von Knoten und Relationen in Wissensgraphen ist die Verwendung des Skalarprodukts von Embeddings, um zu bestimmen, ob eine Beziehung zwischen diesen Knoten bestehen sollte. Ein solcher Ansatz ist DistMult [Yan+14]. In DistMult wird jede Relation durch eine Diagonalmatrix repräsentiert. Dadurch ergibt sich die folgende Scoring-Funktion für einen Wissensgraph $G(V,E,L)$:

$$p(h,l,t) = ||v_h^T W_l v_t||_2, < h,l,t > \in E \tag{2.6}$$

Dabei repräsentiert die Diagonalmatrix W_l die Relation l. Da die Repräsentation einer Relation eine Diagonalmatrix ist, können einige Aspekte mit dieser Darstellungsart nicht erfasst werden. Beispielsweise sind antisymmetrische Relationen (wie zum Beispiel die Relation $\leq$ für natürliche Zahlen) mit DistMult nicht abbildbar. Der Grund hierfür ist, dass immer $||v_h^T W_l v_t|| = ||v_t^T W_l v_h||$ gilt. Es ist zum Beispiel möglich die Einschränkung, dass W_l eine Diagonalmatrix ist, fallen zu lassen. RESCAL [Nic+11] verwendet diesen Ansatz, allerdings wird dadurch deutlich erhöhter Trainingsaufwand verursacht [Ali+20].

Eine andere Methode, antisymmetrische Relationen zu repräsentieren, ist ComplEx [Tro+16]. ComplEx nutzt für die Scoring-Funktion das Hadamard-Produkt auf komplexen Vektoren. Das Hadamard-Produkt bildet die elementweise Multiplikation der Matritzen ab:

Definition 2.3.2 (Hadamard-Produkt) *Das Hadamard-Produkt zweier Matritzen* $A, B \in \mathbb{R}^{n \times m}$ *ist definiert als*

$$A \circ B = \begin{pmatrix} a_{1,1} \cdot b_{1,1} & \dots & a_{1,n} \cdot b_{1,n} \\ \dots & \dots & \dots \\ a_{m,1} \cdot b_{m,1} & \dots & a_{n,m} \cdot b_{n,m} \end{pmatrix}$$

Mit diesem Produkt kann auch DistMult dargestellt werden. Dieses Produkt wird äquivalent auf komplexen Vektoren verwendet. Dazu wird die folgende Scoring-Funktion für den Wissensgraphen $G(V, E, L)$ verwendet:

$$p(h, l, t) = Re(v_h \circ v_l \circ v_t)), < h, l, t > \in E \tag{2.7}$$

Dabei sind v_h, v_l und v_t komplexe Vektoren die jeweils v, l und t repräsentieren. Auf komplexen Räumen ist das Hadamard-Produkt nicht kommutativ. Dem entsprechend gilt die zentrale Einschränkung von DistMult (die fehlende Abbildbarkeit von anitsymmetrischen Relationen) für ComplEx nicht.

2.3.2 Klassifikationsverfahren

In dieser Arbeit werden binäre Klassifikationsverfahren verwendet, um die Anpassung von Ontologie-Alignments zu unterstützen. Klassifikationsverfahren werden an dieser Stelle als die Verfahren behandelt, die bei gegebenen Features einem konkreten Beispiel (zum Beispiel einem

Knoten im Graph) eine Klasse zuordnen können. In dieser Arbeit werden nur binäre Klassifikationsverfahren eingesetzt, die überwacht gelernt werden.

Definition 2.3.3 (Klassifikationsverfahren) *Eine Klassifikationsfunktion $\lambda : \mathbb{R}^n \to L$ ordnet Objekten, die durch Vektoren repräsentiert werden, eine Klasse zu.*

Dabei wird die Funktion λ anhand von Beispielen aus einer Trainingsmenge mit Beispielen gelernt, deren Klasse bekannt ist.

Ein häufig verwendetes Beispiel für einen Klassifikationstask für Graphen nutzt den AIFB-Datensatz [BS07]. Dieser Datensatz beschreibt die Projekt- und Mitarbeiterstruktur des Instituts für Angewandte Informatik und Formale Beschreibungsverfahren des Karlsruher Instituts für Technologie als Ontologie. Die Klassifikation, die durchgeführt werden soll, ist die Vorhersage, welche Personen welcher Forschungsgruppe zugeordnet werden. Die Menge L repräsentiert also bei Anwendung von Definition 2.3.3 die jeweiligen Forschungsgruppen und die Personen werden durch die Menge $\mathbb{R}^n$ repräsentiert. Als Repräsentation können zum Beispiel die in Abschnitt 2.3.1 vorgestellten Graph Embeddings verwendet werden. In Listing 2.2 ist ein in N3-Notation formulierter Ausschnitt dieses Datensatzes für einen Mitarbeiter dargestellt. Es ist zu sehen, dass für jeden Mitarbeiter neben Kontaktinformationen(Zeilen 2-7) Publikationen (Zeilen 8-14, hier nur verkürzt dargestellt) und Projekte (Zeilen 15-18, ebenfalls verkürzt dargestellt) aufgeführt werden. Diese Publikationen und Projekte erzeugen ein Netz zwischen den Mitarbeitern. Da es wahrscheinlich ist, dass Personen, die gemeinsam an Papern oder Projekten arbeiten, in der gleichen Forschungsgruppe arbeiten, lässt sich anhand dieser Informationen mit einer gewissen Genauigkeit das Klassifikationsproblem bearbeiten.

```
1  <http://www.aifb.uni-karlsruhe.de/Personen/viewPersonOWL/id2023instance>      a
       :Person, PHDStudent;
2    :affiliation  <http://www.aifb.uni-karlsruhe.de/Forschungsgruppen/
         viewForschungsgruppeOWL/id3instance>;
3    :fax "+49 (721) 693717"^^<http://www.w3.org/2001/XMLSchema#string>;
4    :homepage "http://www.aifb.uni-karlsruhe.de/WBS/pha/"^^<http://www.w3.org
         /2001/XMLSchema#string>;
5    :name "Peter Haase"^^<http://www.w3.org/2001/XMLSchema#string>;
6    :phone "+49 (721) 608 3705"^^<http://www.w3.org/2001/XMLSchema#string>;
7    :photo "http://www.aifb.uni-karlsruhe.de/Personen/Bilder/U1p2l3o4a5d2023"^^<
         http://www.w3.org/2001/XMLSchema#string>;
8    :publication <http://www.aifb.uni-karlsruhe.de/Publikationen/
         viewPublikationOWL/id1003instance>,
9      <http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/
           id1006instance>,
```

```
10      <http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/
            id1007instance>,
11      <http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/
            id1024instance>,
12      <http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/
            id1025instance>,
13      <http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/
            id1038instance>,
14      ...
15  :worksAtProject <http://www.aifb.uni-karlsruhe.de/Projekte/viewProjektOWL/
        id42instance>,
16      <http://www.aifb.uni-karlsruhe.de/Projekte/viewProjektOWL/id49instance>,
17      <http://www.aifb.uni-karlsruhe.de/Projekte/viewProjektOWL/id51instance>,
18      ...
```

Listing 2.2: Ausschnitt aus AIFB

In Abbildung 2.3 wird eine Hauptkomponentenanalyse von Embeddings der Personen aus dem AIFB-Datensatz dargestellt. Die Zugehörigkeiten zu Arbeitsgruppen (also die Klassen) werden in der Abbildung durch Farben dargestellt. Auch in der auf zwei Dimensionen reduzierten Darstellung lässt sich eine Trennung zwischen den verschiedenen Klassen beobachten. Eine automatisierte Klassifikation lässt sich zum Beispiel durch die in der Abbildung eingezeichneten Geraden erreichen. Anhand dieser Geraden lässt sich eine relativ genaue Zuordnung der Punkte zu den farblich markierten Klassen erreichen, da in den Flächen zwischen den Geraden fast nur Punkte einer Farbe zu finden sind. Ein einfaches Verfahren zum automatischen Zuordnen von Punkten im Raum zu Klassen könnte zum Beispiel in der Suche nach Geraden bestehen, die den Raum so aufteilen, dass der Fehler bei der Zuordnung möglichst gering wird. Im weiteren Verlauf dieses Abschnitts werden ausgefeiltere Verfahren zum Lernen der automatisierten Zuordnung von Beispielen zu Klassen vorgestellt.

In diesem Abschnitt werden sowohl die in dieser Arbeit verwendeten Klassifikationsverfahren als auch Metriken, die bei der Evaluation dieser Verfahren eingesetzt werden können, vorgestellt. Dabei wird auf die Verfahren nur kurz eingegangen. Alle hier verwendeten Verfahren werden in [RN10] detailliert beschrieben, hier wird die grobe Funktionsweise der Verfahren nur kurz angerissen.

Die Verfahren stammen aus den Verfahrensklassen (1) Regression, (2) Naive Bayes, (3) Nearest Neighbour-Verfahren, (4) Entscheidungsbäume, (5) Support Vector Machines und (6) Neuronale Netze.

(1) **Regression** Eigentlich sind Regressionsverfahren grundsätzlich andere Verfahren als

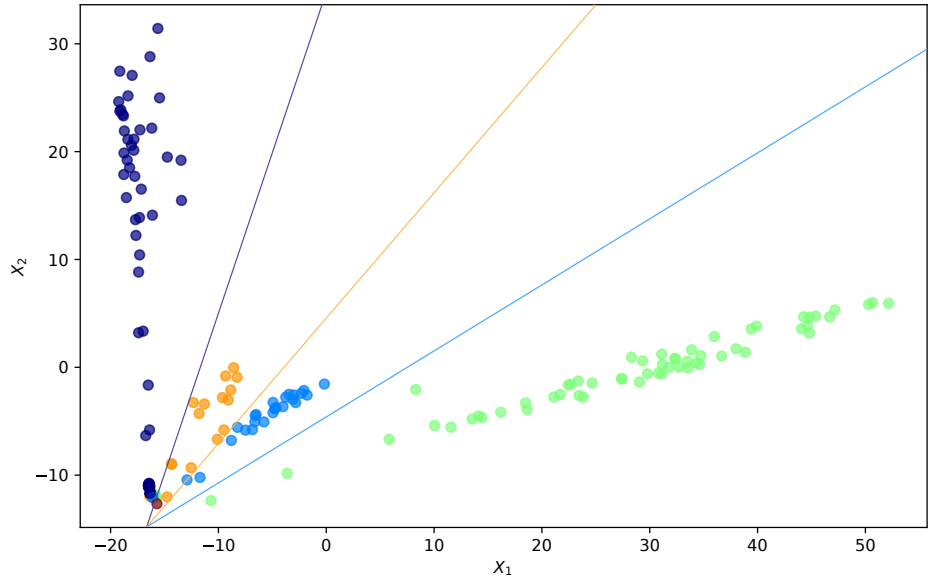

Abbildung 2.3: Hauptkomponentenanalyse von Embeddings des AIFB-Datensatzes

Klassifikationsverfahren. Während Klassifikationsverfahren üblicherweise versuchen, Beispielen ein Label zuzuweisen, versuchen Regressionsverfahren einem Beispiel einen Score zuzuweisen. Bei manchen Klassifikationsverfahren kann dieser Score allerdings unter Verwendung eines Thresholds sinnvoll zur Zuweisung eines Labels verwendet werden. Bei der logistischen Regression wird die logistische Funktion als Regressionsfunktion verwendet. Die logistische Funktion ist

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

Wenn als Eingabe Vektoren verwendet und als Labels die Zahlen 0 und 1 genutzt werden, kann diese Funktion als Klassifikationsfunktion genutzt werden: Die Funktion wird auf das Skalarprodukt eines Vektors $\boldsymbol{w}$ und dem Feature-Vektor des Beispiels $\boldsymbol{x}$ angewandt. $\boldsymbol{w}$ ist der Gewichtsvektor und wird mithilfe des Gradientenabstiegsverfahren gelernt. Dieses Verfahren wird auch in dieser Arbeit eingesetzt.

(2) **Naïve Bayes** Das Naïve-Bayes-Verfahren basiert auf der Bayesschen Regel zur bedingten Wahrscheinlichkeit. Wenn angenommen wird, dass alle Features von einander unabhängig

sind (was üblicherweise nicht der Fall ist, darum wird das Verfahren als „Naïve“ bezeichnet) kann die Wahrscheinlichkeit, dass bei gegebenen Features $x_1 ... x_n$ ein Beispiel zur Klasse C gehört, durch die folgende Formel angegeben werden.

$$P(C|x_1, ...x_n) = \alpha P(C) \prod_{i=1}^{n} P(x_i|C) \tag{2.9}$$

α ist dabei ein Faktor, der konstant ist. Das Naïve-Bayes-Verfahren hat zwar eine eingeschränkte Ausdrucksstärke, wird aber in der Praxis häufig verwendet, da es sehr einfach angewandt werden kann [RN10, s. 808].

(3) **Nearest Neighbour** Nearest-Neighbour-Verfahren basieren auf der Idee, dass sich aus den Features ein Raum ergibt, in dem die Distanz zwischen Beispielen bestimmt werden kann. Um Klassifikation durchzuführen, werden die k nächsten Nachbarn eines Beispiels betrachtet. Die Klasse, die unter diesen Nachbarn am häufigsten vorkommt, wird dann auch dem vorliegenden Beispiel zugeordnet. Als Distanzmaß kann zum Beispiel die euklidische Distanz im Feature-Raum verwendet werden.

(4) **Entscheidungsbäume** Entscheidungsbäume repräsentieren die zu lernende Klassifikation in der Form eines Baums. Entscheidungen werden in den jeweiligen Knoten des Baumes bezogen auf ein Feature getroffen. Um die Entscheidungsbäume automatisiert erstellen zu können, werden Schwellwerte für die jeweiligen Features gewählt und die Entropie der durch die Entscheidung entstehenden Aufteilung beurteilt. Mit der Entropie lässt sich die Unsicherheit einer Zufallsvariable messen. Üblicherweise werden neue Entscheidungsknoten so gewählt, dass die Entropie am stärksten sinkt. [RN10, s. 703 ff] In dieser Arbeit werden zwei Entscheidungsbaum-Verfahren verwendet: CART [Bre+84] und Random Forests [Ho95]. CART lernt einen einzelnen Entscheidungsbaum, wobei ein Mechanismus verwendet wird, um den Baum möglichst kompakt zu halten. Dieser Mechanismus wird als Pruning bezeichnet. Random Forests erstellen ein Ensemble von Bäumen, die kleiner sind, als solche Bäume, die mit CART erstellt werden. Ein Pruning ist wegen der kleineren Bäume nicht notwendig.

(5) **Support Vector Machines** Support Vector Machines basieren auf zwei Konzpeten: der Idee eines *Max Margin Separators* und des *Kernel Tricks*. Ein Max Margin Separator arbeitet so, dass bei binärer Klassifikation eine Grenze durch den Feature-Raum gezogen wird, die einerseits möglichst alle Beispiele korrekt klassifiziert und andererseits so gestaltet ist, dass die Begrenzung möglichst weit von Beispielen beider Klassen entfernt ist. Diese Grenze wird durch eine Hyperebene bestimmt. Um auch nicht durch eine Hyperebene beschreibbare Probleme lösen zu können, wird der Kernel Trick verwendet. Beim Kernel Trick wird auf den Feature-

Raum eine Transformation angewandt, die auf einer Kernel-Funktion basiert. So werden die Werte in einen Raum transformiert, in dem die Daten mit einer Hyperebene separiert werden können. Typische Kernel-Funktionen sind lineare Kernel, polynomische Kernel und Kernel, die auf der radialen Basis Funktion (RBF) basieren. RBF ist die quadrierte euklidische Distanz zwischen zwei Vektoren.

(6) **Neuronale Netze** Neuronale Netze sind verallgemeinerte Erweiterungen von Regressionsmethoden. Ein neuronales Netz besteht aus mehreren Neuronen. Ein Neuron besitzt mehrere gewichtete Eingaben und eine Aktivierungsfunktion. Die Ausgabe des Neurons wird als Aktivierung bezeichnet. Die Aktivierung eines Neurons kann dann als Eingabe eines anderen Neurons verwendet werden oder als Ausgabe des sich durch die Verknüpfung ergebenden Netzes. Die Aktivierung eines Netzes wird wie folgt berechnet:

$$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{2.10}$$

dabei ist a_j die Aktivierung des $j-$ten Neurons, $g : \mathbb{R}^n \to \mathbb{R}$ die Aktivierungsfunktion, n die Anzahl der Gewichte je Neuron und $w_{i,j}$ das Gewicht, dass das Neuron i mit dem Neuron j verbindet. Wenn nur ein Layer verwendet wird (also die Eingabe aller Neuronen die Eingabe des Netzes ist und die Ausgabe aller Neuronen die Ausgabe des Netzes) wird das Netz als Perceptron bezeichnet. Die Gewichte des Perceptrons können wie auch bei linearer Regression mit einem Gradientenabstiegsverfahren gelernt werden. Werden die Ausgaben eines Neurons mit der Eingabe eines anderen Neurons verbunden, können die Neuronen in sogenannten Layern betrachtet werden. das 0-te Layer sind alle Neuronen, die direkt mit der Eingabe verbunden sind, das 1-te Layer sind alle Neuronen, die direkt mit dem 0-ten Layer verbunden ist, etc. Wenn alle Neuronen des n-ten Layers mit allen Neuronen des $n+1$-ten Layers verbunden sind, spricht man von einem Multilayer Perceptron. Beim Training stellt sich dadurch die Frage, wie ein Gradient an den Neuronen berechnet werden kann, die nicht im letzten Layer sind. Dazu wird das Backpropagation-Verfahren verwendet. Backpropagation erlaubt die Berechnung eines Neuron-spezifischen Fehlers Δ_j mit der folgenden Formel:

$$\Delta_j = g'(in_j) \sum_k w_{j,k} \Delta_k \tag{2.11}$$

Dabei ist in_j die Eingabe des j-ten Neurons, $g'()$ die erste Ableitung der Aktivierungsfunktion und $w_{j,k}$ die Gewichte, die die Ausgabe des j-ten Neurons mit dem k-ten Neuron

verbinden.

2.3.3 Bewertung der Ergebnisse von Klassifikationsverfahren

Um die Güte von Klassifikationsverfahren bei der Anwendung auf konkrete Eingaben zu bewerten, werden Metriken benötigt. Diese Metriken werden auf Test-Mengen berechnet, für die die korrekten Ergebnisse bekannt sind. In dieser Arbeit werden nur binäre Klassifikationsverfahren betrachtet. Bei der Beurteilung der Klassifikation wird untersucht, wie gut der Klassifikator in der Lage ist, die Zugehörigkeit zu einer der Klassen zu erkennen. In diesem Kontext bezeichnet TP die richtig positiven, FP die falsch positiven, TN die korrekt negativen und FN die falsch negativen Ergebnisse des Klassifikators.

Accuracy misst die Genauigkeit eines Klassifikators, also wie häufig richtig klassifiziert wird, gemessen an der Anzahl der Vorhersagen. Dem entsprechend ist accuracy definiert als

Definition 2.3.4 (Accuracy)

$$acc = \frac{\#TP + \#TN}{\#TP + \#FP + \#TN + \#FP}$$

.

Diese Metrik gibt also einen Aufschluss darüber, wie die Leistung des Klassifikators bezogen auf beide Klassen ist. Precision bezeichnet die Genauigkeit bezogen nur auf eine Klasse. Sie setzt die richtig positiven ins Verhältnis zu allen positiven Ergebnissen, sie wird also als

Definition 2.3.5 (Precision)

$$prec = \frac{\#TP}{\#TP + \#FP}$$

definiert. Der Recall gibt an, wie groß der Anteil der Elemente einer Klasse ist, die gefunden werden können. Berechnet wird diese Metrik mit

Definition 2.3.6 (Recall)

$$rec = \frac{\#TP}{\#TP + \#FN}$$

.

Precision und Recall sagen für sich genommen noch nicht viel aus: ein Verfahren, dass kaum Elemente positiv klassifiziert, kann eine sehr hohe Precision erreichen und bei einem Verfahren,

dass alle Beispiele positiv klassifiziert, wird einen Recall von 1 gemessen. Um diese Extreme jeweils auszugleichen wird der f1-score verwendet, der den harmonischen Mittelwert zwischen Precision und Recall bildet. Der f1-score wird berechnet als

Definition 2.3.7 (f1-Score)

$$f1 = 2 * \frac{prec * rec}{prec + rec}$$

Intern werden Klassifikationsverfahrens häufig nicht mit einer rein binären Ausgabe umgesetzt (also „gehört zu Klasse x“ oder „gehört nicht zu Klasse x“), sondern produzieren einen Score, der die Wahrscheinlichkeit repräsentieren soll, mit der ein betrachtetes Beispiel zu einer Klasse gehört. Dieser Score wird dann üblicherweise mit einem Schwellwert in eine Entscheidung des Klassifikators umgewandelt. Dem entsprechend können Klassifikationsverfahren noch detaillierter darauf untersucht werden. Es ist zum Beispiel relevant zu betrachten, mit welchem Score Fehleinschätzungen des Klassifikators abgegeben werden. Diese Betrachtung kann detailliertere Einblicke geben, wie gut ein Verfahren in der Lage ist, Klassen zu unterscheiden.

Eine Metrik, die nach diesen Kriterien Klassifikatoren bewertet, ist die Fläche unter der Receiver-Operating-Characteristics-Kurve (ROC). Häufig wird dieser Wert mit roc_auc abgekürzt. Die ROC-Kurve wird so gebildet, dass die Falsch-Positiv-Rate ($\frac{\#FP}{\#FP+\#TN}$) auf der X-Achse aufgetragen wird und die True-Positive-Rate ($\frac{\#TP}{\#TP+\#FN}$) auf der Y-Achse. In dem Graph werden dann die entsprechenden Raten bei variierendem Threshold des Klassifikators eingetragen. Die Fläche unter der dabei entstehenden Kurve ist der Wert, der von dieser Metrik zurückgegeben wird.

Eine andere Möglichkeit, diese Kriterien mit einzubeziehen, ist die Fläche unter der Precision-Recall-Kurve, was auch als average Precision [Zhu04] bezeichnet wird. Wenn $p(r)$ die Precision bei Recall r ist, ist die Average Precision

Definition 2.3.8 (Average Precision)

$$avg\_prec = \int_0^1 p(r)dr$$

.

In der Praxis wird dies üblicherweise über eine nach Score sortierte Ergebnisliste berechnet. Wenn $p(i)$ die Precision der ersten i Ergebnisse und $\Delta r(i)$ die Differenz zwischen dem Recall der ersten i Elemente und der ersten $i-1$ Elemente ist, dann ist der in der Praxis verwendete

Wert für Average Precision:

$$avg\_prec = \sum_{i=1}^{n} p(i)\Delta r(i) \tag{2.12}$$

Alle diese Metriken sind so gestaltet, dass höhere Werte immer das wünschenswerte Ergebnis darstellen. Allerdings kann kein Wert wirklich unabhängig vom betrachteten Datensatz und der Bewertung von anderen Verfahren betrachtet werden. Eine Accuracy von 99% kann zum Beispiel auf einem Datensatz, bei dem 99% der Beispiele zu einer Klasse gehören sehr einfach erreicht werden: Das Klassifikationsverfahren schlägt immer die Klasse vor, zu der die meisten Beispiele gehören.

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden Ansätze aus der Literatur vorgestellt, die ähnliche Probleme behandeln wie diese Arbeit. Außerdem wird untersucht, welche Aspekte bisher noch nicht betrachtet wurden und entsprechend im weiteren Verlauf der Arbeit betrachtet werden. Die Aufgabe, Verbindungen zwischen Schemata an Veränderungen anzupassen, ist bereits im Bereich der Datenbank- und XML-Schemata untersucht worden und wird dort unter dem Begriff „Mapping Adaption Problem“ betrachtet [Vel+03; YP05]. Da die in diesem Bereich verwendeten Ansätze später auf Ontologien angewendet werden, werden sie hier mithilfe der gleichen formalen Notation betrachtet, wie die Ansätze, die Ontologie-Alignments betrachten. Die Begriffe Alignments und Mappings werden in der Literatur häufig synonym verwendet. In dieser Arbeit wird in Anlehnung an [ES07] der Begriff „Alignment“ verwendet, da für den Prozess des Findens von Alignments häufig die Begriffe „Mapping“ [Har+19] oder „Matching“ [Thi+19] verwendet werden.

In der Literatur zu Ontologie-Alignments werden die Ontologien häufig als Mengen von Konzepten betrachtet. Diese Sichtweise wird hier auch verwendet, um die verwandten Arbeiten darzustellen, im weiteren Verlauf der Arbeit (Abschnitt 4.3) wird eine gleichwertige Definition vorgestellt, die Ontologien als Graphen betrachtet. Da die Betrachtung der Ontologien als Graphen zur Betrachtung der verwandten Arbeiten nur zusätzlichen formalen Aufwand bedeuten würde und nicht zum Verständnis beitragen würde, wird hier die auf Mengen basierende Definition verwendet. Diese Definition ist eine Kurzschreibweise für die in Definition 2.2.2 vorgestellte Definition eines Alignments. In [Gro+13; Jur16] wird ein Alignment zwischen den Ontologien $\mathcal{O}_1$ und $\mathcal{O}_2$ als Menge von Tupeln aus Konzepten der beiden Ontologien und einem Verbindungstyp betrachtet:

$$M_{\mathcal{O}_1,\mathcal{O}_2} = \{(c_1, c_2, semType) | c_1 \in \mathcal{O}_1, c_2 \in \mathcal{O}_2, semType \in \{\equiv, \leq, \geq\}\}$$

Elemente der Menge $M_{\mathcal{O}_1,\mathcal{O}_2}$ werden Mapping-Einträge oder Alignment-Statements genannt. $\equiv, \leq$ und $\geq$ stehen in diesem Zusammenhang für die Äquivalenz von zwei Konzepten, eine Unterklassen-Beziehung und eine Oberklassen-Beziehung.

Die veränderte Version einer Ontologie $\mathcal{O}$ wird üblicherweise als $\mathcal{O}'$ bezeichnet. Das Mapping-Adaption-Problem für zwei Ontologien $\mathcal{O}_1$ und $\mathcal{O}_2$, die durch das Alignment $M_{\mathcal{O}_1,\mathcal{O}_2}$ verbunden sind, besteht darin, ein neues Alignment $M'_{\mathcal{O}_1,\mathcal{O}'_2}$ zu finden, wenn $\mathcal{O}_2$ sich zu $\mathcal{O}'_2$ verändert.

Die verwandten Arbeiten werden in den folgenden Abschnitten vorgestellt: Zunächst werden Ansätze vorgestellt, die ebenfalls den Einfluss von Veränderungen auf Alignments betrachten (Abschnitt 3.1). Diese Ansätze verwenden manuell festgelegte Regeln, um die Reaktion auf Veränderungen abzubilden. Verfahren des maschinellen Lernens werden nicht betrachtet. Es gibt allerdings Ansätze, die Verfahren des maschinellen Lernens im Zusammenhang mit der Vorhersage von Veränderungen in den Ontologien selbst betrachten (Abschnitt 3.2). Diese Verfahren nutzen meist lokale Features von Ontologieknoten, um an diesen Veränderungen vorherzusagen. Dazu werden nur händisch festgelegt Features verwendet, die Verwendung von gelernten Repräsentationen der Ontologieknoten wird nicht betrachtet. Dieser Aspekt wird von Ansätzen zur automatisierten Erstellung und Vervollständigung von Alignments mit einbezogen, die Verfahren zum Lernen von Repräsentation einsetzen (Abschnitt 2.3.1). Diese Ansätze betrachten keine Veränderungen. Im Anschluss an die Vorstellung der verwandten Arbeiten wird dargestellt, welche Aspekte bisherige Ansätze aus der Literatur außer Acht gelassen haben und in dieser Arbeit betrachtet werden sollen (Abschnitt 3.4).

3.1 Alignments und Veränderungen

Dieser Abschnitt stellt Verfahren vor, die sich direkt mit der Anpassung von Alignments an Veränderungen in Ontologien beschäftigen. Zunächst werden Arbeiten vorgestellt, die sich mit der verwandten Themenstellung der Anpassung von Mappings zwischen XML- und Datenbank-Schemata beschäftigen. Darauf folgt eine Vorstellung von Ansätzen, die die Methoden dieser Ansätze auf biomedizinische Ontologien anwenden und die Vorstellung eines weiteren Ansatzes, der auf Beispielen der OAEI arbeitet. Noch nicht betrachtet werden Verfahren des maschinellen Lernens oder Anwendungen auf Ontologien aus anderen Domänen.

Ein zentraler Ansatz zur Anpassung von Mappings im Bereich der Datenbanken wurde in [Vel+03] vorgestellt. Für jede Veränderung innerhalb eines Schemas werden Regeln angewandt. Die jeweiligen Regeln werden durch Veränderungsmuster definiert, auf die jeweils eine Reaktion erfolgt. Dieser Ansatz wird in der Literatur als *inkrementeller Ansatz* bezeichnet [Rei+15b]. Ein weiterer Ansatz [YP05] basiert darauf, ein neues Mapping $M'_{\mathcal{O}_1,\mathcal{O}'_2}$ automatisiert zu erzeugen, in

dem dieses Mapping als eine *Komposition des vorhandenen Mappings* $M_{\mathcal{O}_1,\mathcal{O}_2}$ und eines neuen Mappings $M^+_{\mathcal{O}_2,\mathcal{O}'_2}$ dargestellt wird. $M^+_{\mathcal{O}_2,\mathcal{O}'_2}$ ist dabei ein Versions-Mapping, das Entitäten aus der alten Ontologie auf die neue Ontologie abbildet. Dem entsprechend wird dieses Vorgehen als *Kompositioneller Ansatz* beschrieben [Rei+15b]. Beide Ansätze betrachten dabei nicht die Semantik der Schemata, sondern arbeiten nur auf der Syntax oder auf Hintergrundwissen über die Klassen, die mit dem Schema abgebildet werden.

Auf diese Arbeiten wurde unter anderem in [Gro+13] aufgebaut: Es wurde untersucht, wie diese Ansätze auf biomedizinischen Daten angewandt werden können. Zur Anwendung des inkrementellen Ansatzes wurde das Werkzeug COnto-Diff [Har+13] erstellt, das verschiedene Arten von Veränderungen in Ontologien identifizieren kann und auf biomedizinische Ontologien ausgerichtet ist. Je nach Zuordnung einer Ontologie-Veränderung zu einer Klasse im Veränderungsmodell werden Regeln zur Anpassung des Ontologie-Alignments angewandt. Der Kompositionelle Ansatz berechnet das Mapping zwischen den verschiedenen Ontologieversionen $M^+_{\mathcal{O}_2,\mathcal{O}'_2}$ mithilfe von gegebenen Ähnlichkeiten der Knoten zwischen den Versionen. Aufbauend darauf wird die Komposition $M'_{\mathcal{O}_1,\mathcal{O}'_2}$ erzeugt. Bei der Evaluation wurde so vorgegangen, dass mit dem Verfahren auf einem historischen Biomedizinischen Datensatz ein neues Alignment vorgeschlagenen wurde und dieses vorgeschlagene Alignment mit dem in der Historie tatsächlich verwendeten Alignments verglichen wurde. In der Untersuchung konnte gezeigt werden, dass der inkrementelle Ansatz beim Vorschlagen von Änderungen an Alignments die besten Ergebnisse hinsichtlich Precision, Recall und F-Measure erzeugen konnte. Auf dem betrachteten Datensatz ist der Anteil von nicht von Veränderungen betroffenen Alignment-Statements sehr hoch, auf einem Teil des Datensatzes ist der Anteil 94%, auf einem weniger stabilen Teil 80%, was bei der Interpretation der Ergebnisse berücksichtigt werden muss. Auf dem weniger stabilen Teildatensatz konnte eine Precision von 90%, ein Recall von 98% und F-Measure von 94% erziehlt werden. Da beim stabilen Teil des Datensatzes der Anteil an unveränderten Alignment-Statements sehr hoch ist, sind Precision, Recall und F-Measure nicht besonders aussagekräftig.

In [dos+15] wurde der inkrementelle Ansatz um Heuristiken erweitert, mit deren Hilfe weitere Veränderungen am Alignment vorgeschlagen werden können. Diese Heuristiken basieren auf Veränderungsmustern, die durch eine statistische Untersuchung auf biomedizinischen Datensätzen in [DR+14] gefunden wurden. Diese Heuristiken wurden mit den Regeln auf der Basis von COnto-Diff aus [Gro+13] kombiniert. Bei einer Untersuchung auf einem ähnlichen Datensatz wie [Gro+13] konnten hinsichtlich Precision und Recall bessere Ergebnisse erzielt werden, als wenn ausschließlich Regeln verwendet werden. Auf einem Datensatz, bei dem der Anteil der nicht von Veränderungen betroffenen Alignment-Statements 59% ist, konnte eine Precision

von 99%, ein Recall von 80% und ein F-Measure von 88% erreicht werden. Dem entsprechend ist die Differenz zwischen Precision und dem Anteil der nicht von Veränderungen betroffenen Alignment-Statements größer, als für den von [Gro+13] vorgestellten Ansatz.

Diese in [dos+15] und [Gro+13] vorgestellten Arbeiten zeigen, dass das Mapping-Adaption-Problem bereits mit Regeln und Heuristiken bearbeitet werden kann. Allerdings ist ein weiteres interessantes Ergebnis, das auf diesen Datensätzen ein sehr großer Teil des Alignments nicht von Veränderungen betroffen ist. Es ist aber nicht klar, wie im Vorhinein bestimmt werden kann, welche Knoten eines Alignments nicht von Veränderungen betroffen sind. Dem entsprechend kann es auch eine wichtige Aufgabe sein, herauszufinden, welche Einträge des Alignments angepasst werden müssen und bei welchen Einträgen keine Anpassung notwendig ist.

Ein weiterer Ansatz [KP20] schlägt ein anderes Vorgehen vor. Zunächst werden für gelöschte Knoten die Alignments entfernt. Für hinzugefügte Knoten wird ein Ansatz zur automatisierten Erstellung von Ontologie-Alignments (wie in Abschnitt 2.2) genutzt. Dieser Ansatz wird auf Beispielen der OAEI (siehe Abschnitt 2.2.2) evaluiert, die durch zufällige Veränderungsoperationen angepasst wurden. So stehen zu allen Beispielen zwei Ontologien zur Verfügung, eine veränderte und eine unveränderte. Dieser Datensatz ist sehr klein und die Art der Veränderungen wird nicht näher beschrieben, dem entsprechend sind die Ergebnisse nicht mit den Ergebnissen der anderen Ansätze vergleichbar. Der zentrale Vorteil dieses Ansatzes besteht in der Rechenzeitersparnis, da nur ein kleiner Teil der Ontologien betrachtet werden muss.

Wichtige Informationen, die außer Acht gelassen werden, sind unter anderem die Betrachtung von Axiomen und die Betrachtung der Semantik von Veränderungen [Gro+16]. Beide Aspekte lassen sich in den Inferenzen der Ontologien betrachten, da Semantik und Axiome beide in den Inferenzen beobachtet werden können. Diese Analysen führen zu einem weiteren Bereich, der von der Forschung bisher nicht betrachtet wurden: Es wurde bisher nicht untersucht, wie Inferenzen in den Veränderungsprozess von mit Alignments verbundenen Ontologien einbezogen werden können.

Ein weiterer wichtiger Aspekt, der in der Literatur bisher nicht untersucht wurde, ist die durch maschinelles Lernen unterstütze Anpassung von Alignments. In [dos+15] wurden mithilfe einer statistischen Analyse Muster in den Veränderungen von Alignments gefunden. Diese Muster wurden dann verwendet, um den Veränderungsprozess automatisiert zu unterstützen. Es ist denkbar, dass ein solches Verfahren auch automatisiert arbeiten könnte: Auf aus der Historie bereits bekannter Versionssprüngen könnte mithilfe von Methoden des maschinellen Lernens ein Modell trainiert werden, dass die Anpassung von Alignments automatisiert unterstützt. Ein solches Verfahren wurde bisher in der Literatur noch nicht betrachtet.

Die Einbindung von Informationen zu der Position eines Knotens in der Graphstruktur wird

bisher in der Literatur noch nicht mit einbezogen. Zwar sind die von [Gro+13] und [dos+15] verwendeten Informationen teilweise auch struktureller Natur, allerdings werden nur kleine Strukturen in der direkten Nachbarschaft eines veränderten Knoten betrachtet. Es ist denkbar, dass je nach der Position eines Knoten im gesamten Graphen Veränderungen größere Einflüsse auf das Alignment haben. Trotzdem wurde dieser Aspekt bisher nicht untersucht. Dazu müsste die Graphstruktur als Ganzes betrachtet werden. Die vorliegende Arbeit widmet sich unter anderem der Frage, ob auch strukturelle Aspekte mit Bezug zum gesamten Graph in die Anpassung von Alignments an Ontologie-Veränderungen einbezogen werden können.

Außerdem werden bisher vor allem Datensätze aus der Biomedizin betrachtet, oder es wird auf Datensätzen gearbeitet, die zur Evaluation von automatisierten Ansätzen zur Erstellung von Alignments verwendet werden und durch zufällige Veränderungen angepasst wurden. Ontologien aus anderen Domänen werden bisher nicht verwendet. Dem entsprechend ist die Anwendbarkeit solcher Ansätze auf andere Domänen noch unklar.

3.2 Veränderungen in Ontologien und maschinelles Lernen

Es existieren bereits mehrere Ansätze, die maschinelles Lernen zur Vorhersage von Veränderungen an Ontologien verwenden, diese werden im folgenden diskutiert. Die meisten Ansätze arbeiten mithilfe von händisch ausgewählten Features. Verfahren zum automatisierten Lernen von Repräsentationen werden nicht eingesetzt.

Zur Betrachtung von Veränderungen in Ontologien mit Methoden des maschinellen Lernens gibt es einige Vorarbeiten aus dem Bereich der biomedizinischen Ontologien: In [PC12] wurde eine Methode vorgestellt, mit der Ergänzungen zu bestehenden Ontologien vorhergesagt werden sollen. Das konkrete Ziel der Methode ist vor allem, Bereiche der Ontologie vorherzusagen, die wahrscheinlich in der nächsten Versionsveränderung angepasst werden. Dazu werden strukturelle Informationen, wie die Anzahl von Unter- und Oberklassen oder Anzahl an Annotationen oder Informationen aus Literaturdatenbanken, wie zum Beispiel, wie oft das Element der Ontologie in der Literatur referenziert wurde, mit einbezogen. Auf diese Informationen werden Verfahren des maschinellen Lernens, unter anderem Support Vector Machines, Multilayer Perceptrons und das Naïve-Bayes-Verfahren angewandt. Auf einem realen Datensatz (Versionssprünge der Gene Ontology [Con04]) konnte gezeigt werden, dass ein großer Teil der zu verändernden Knoten korrekt vorhergesagt werden konnte: es konnte eine Precision von mehr als 80% und ein Recall von ebenfalls mehr als 80% erreicht werden.

Dieser Ansatz wurde in [Tsa+13] erweitert. Neben den strukturellen Informationen und den Literaturinformationen wird hier ein neuer Feature-Typ betrachtet. Dieser neue Feature-Typ

sind temporale Features, die zusammenfassen, wie sich die strukturellen Informationen und Literaturinformationen über die Zeit verändern. Dazu wird auf einen festgelegten Zeitausschnitt bezogen die Unterschiede zwischen den Features am Beginn und Ende des Zeitausschnitts gebildet. Auf diese Informationen werden wie bei [PC12] Verfahren des maschinellen Lernens angewandt. Diese Methode wird auf der MeSH-Ontologie [Nel+01] evaluiert. Es konnte gezeigt werden, dass der Ansatz auf dieser Ontologie eingesetzt werden kann, wenn eine ausreichend lange Historie vorhanden ist: Auf MeSH konnte ein F-Measure von 66,4% erreicht werden.

Beide hier vorgestellten Ansätze sind jeweils auf einen Datensatz begrenzt. Eine Erweiterung auf weitere Datensätze wird in [Car+18] vorgestellt. Dieser Ansatz analysiert, welche Features den größten Beitrag zur Vorhersage von Änderungen hat und evaluiert die automatisierte Vorhersage auf verschiedenen biomedizinischen Datensätzen. Außerdem werden nur Baum-basierte Verfahren des maschinellen Lernens verwendet, damit das trainierte Modell manuell untersucht werden kann. Als die wichtigsten Features werden die Anzahl der Nennungen eines Konzepts in der Literatur, die Anzahl der direkten Kinder eines Knoten und die Anzahl der Annotationen identifiziert. Auf den betrachteten Datensätzen (ICD-9, NCIt, MeSH und Snomed CT) kann eine Accuracy zwischen 68% und 91% erreicht werden. Als mögliche Erweiterung des Ansatzes wird die Anwendung des Verfahrens zur Vorhersage von Alignment-Änderungen gesehen.

Alle diese Ansätze versuchen vorher zu sagen, wie sich eine Ontologie verändern wird. Die Fragestellung in dieser Arbeit ist etwas anders gestaltet: die Ontologie-Veränderungen sind bereits vorgegeben und es muss herausgefunden werden, wie sich das Alignment verändert. Die automatisierte Vorhersage von Alignment-Veränderungen mithilfe von Methoden des maschinellen Lernens wird von [Car+18] als ein vielversprechender Bereich für zukünftige Arbeiten gesehen. Dieser Bereich wird in dieser Arbeit betrachtet.

Eine weitere Gemeinsamkeit der Ansätze sind die betrachteten Features: welche Eigenschaften der zu untersuchenden Graphknoten verwendet werden, wird bei allen Ansätzen manuell festgelegt. Es ist allerdings auch denkbar, diese Eigenschaften zu lernen. Ansätze aus der Sprachmodellierung wie Word2Vec [Mik+13] zeigen, dass auf ausreichend großen Datenmengen für einzelne Worte unüberwacht Repräsentierungen gelernt werden können, die viele wichtige Aspekte dieser Worte abbilden. Solche Verfahren existieren auch für Graphen, zum Beispiel wurde mit RDF2Vec (siehe Abschnitt 2.3.1) ein solcher Ansatz vorgestellt. Ansätze zum unüberwachten Lernen einer Repräsentation wurden im Bereich von Veränderungen und Ontologie-Alignments bisher nicht betrachtet, dieser Bereich wird ebenfalls in dieser Arbeit untersucht.

3.3 Maschinelles Lernen und Ontologie-Alignments

Wie im vorherigen Abschnitt beschrieben, wurden bei der Betrachtung von Veränderungen in Ontologien bisher keine Verfahren zum unüberwachten Lernen von Repräsentationen eingesetzt. Beim Betrachten von Ontologie-Alignments, insbesondere beim Erzeugen von solchen Alignments wurden diese Verfahren bereits eingesetzt. Dabei werden die Ontologien häufig als Graphen betrachtet. Ansätze, die solche Verfahren zum Erweitern oder Erstellen von Alignments einsetzen werden im folgenden vorgestellt.

Ein Verfahren, das auf den in Abschnitt 2.3.1 vorgestellten Verfahren aufbaut wurde in [Che+16] betrachtet. Dieser Ansatz soll Alignments zwischen Wissensgraphen in unterschiedlichen Sprachen vervollständigen. Dazu werden mithilfe von TransE beide Ontologien in jeweils einen Vektorraum abgebildet und dann eine Transformation zwischen diesen Vektorräumen bestimmt. Zur Bestimmung dieser Transformation werden bereits bestehende Alignment-Statements zwischen den beiden Ontologien genutzt. Durch die Transformation lassen sich weitere Alignment-Einträge finden. Eine Bedingung für die Einsetzbarkeit dieses Ansatzes ist also, dass bereits einige Alignments zwischen den Ontolgien bestehen. Eine Evaluation wird auf verschiedenen Sprachen von DBPedia durchgeführt. Es kann gezeigt werden, dass es möglich ist, eine Transformation zwischen den Vektorräumen der Ontologien, die Wissen in verschiedenen Sprachen abbilden, zu lernen. Auf dem Mapping zwischen der englischen und deutschen DBPedia konnte eine Accuracy von 95% erreicht werden. Es existieren noch weitere Ansätze, die auf Embeddings arbeiten, auch diese nutzen Mapping-Informationen, um zwischen den Embedding-Räumen der Ontologien in verschiedenen Sprachen eine Transformation zu finden [Zhu+17; Sun+17; Sun+18].

In [Sun+19] wird ein Verfahren vorgestellt, das ein spezielles auf Graphen ausgerichtetes neuronales Netzwerk einsetzt, um automatisiert ein Alignment zu erzeugen, also die gleiche Aufgabe zu lösen, wie die in Abschnitt 2.2 beschriebenen Verfahren. Das neuronale Netz arbeitet so, dass die Aktivität in der Nachbarschaft eines Graphknoten zusammengefasst wird. So wird versucht, Ähnlichkeiten in der Graphstruktur, die für einen Ontologie-Knoten relevant sind, in den verschiedenen Ontologien zu vergleichen. Dieser Ansatz geht darüber noch hinaus, indem mithilfe eines Attention-Mechanismus auch die Struktur von etwas weiter entfernten Knoten mit einbezogen wird. In einer Evaluation auf einem Datensatz, der aus DBPedia für verschiedenen Sprachen besteht, konnte gezeigt werden, dass dieser Ansatz gewinnbringend eingesetzt werden kann. Auf dem vorher in der Literatur nicht betrachteten Datensatz, der ein Alignment von japanischer und englischer Sprache beinhaltet, konnte eine Accuracy von 82% erreicht werden, wenn der Ansatz zum Repräsentieren der Graphen mit einem Ansatz zur maschinellen Übersetzung kombiniert wird.

Diese vorgestellten Ansätze wurden nur auf Wissensgraphen evaluiert, deren Inhalte fast identisch sind, aber in unterschiedlichen Sprachen notiert wurden. Entsprechend sind diese Graphen strukturell sehr ähnlich. Also ist es auch im Vergleich zu Ontologien, die tatsächlich unterschiedliche Themen behandeln, wie zum Beispiel im Fall von biomedizinischen Ontologien, etwas einfacher ein Alignment auf der Basis von strukturellen Eigenschaften zu finden.

Eine weitere Möglichkeit bei Alignments das Lernen von Repräsentationen mit einzubeziehen wird in [JR+20] betrachtet. Das Ziel dieses Ansatzes ist es, das automatisierte Erstellen von neuen Alignments so in Tasks aufzuteilen, dass diese Tasks effizient isoliert betrachtet werden können. Dabei wird eine Kombination aus Word- und Graph-Embeddings verwendet, um Cluster zu bilden, die dann für die Tasks verwendet werden. Durch eine Evaluation auf einem Biomedizinischen Datensatz kann gezeigt werden, dass die durch die Embeddings erzeugte Aufteilung Vorteile für die Erzeugung eines neuen Alignments bieten kann.

Alle vorgestellten Ansätze verwenden Verfahren zum Lernen von Repräsentationen, um Knoten der betrachteten Ontologien zu verarbeiten. So kann unter anderem auch die Struktur der Ontologien mit einbezogen werden. Nicht betrachtet werden Veränderungen an den Ontologien. Dem entsprechend wird auch nicht betrachtet, ob Methoden zum Lernen von Repräsentationen auf die Vorhersage von Alignment-Änderungen angewandt werden können. Vor-und Nachteile verschiedener Methoden zur Repräsentation von Graph-Knoten mit Bezug auf Alignment-Änderungen wurden entsprechend ebenfalls nicht untersucht. Diese Fragen werden in dieser Arbeit betrachtet.

3.4 Forschungslücken

Aus der Diskussion dieser verwandten Arbeiten ergeben sich einige Bereiche, die bisher in der Literatur nicht betrachtet wurden. Dieser Abschnitt bietet eine Zusammenfassung dieser Forschungslücken. Auf diese Lücken werden im weiteren Verlauf der Arbeit die Forschungsfragen aufgebaut. Im folgenden werden die Aspekte, die in dieser Arbeit untersucht werden aufgeführt. Dazu wird jeweils der Themenbereich genannt, der bisher noch nicht näher in der Literatur betrachtet wurde und kurz beschrieben, welche Aspekte diese Arbeit untersucht.

RG1 Bisher wurde nicht betrachtet, wie die Berechnung von Inferenzen in die Vorhersage von Veränderungen am Alignment mit einbezogen werden können. Aspekte, die die Semantik der Ontologien betrachten, werden bisher meist in manuell erstellten Regeln abgebildet, die festlegen, wie auf Veränderungen reagiert werden soll. In dieser Arbeit soll untersucht werden, ob durch die Betrachtung von Inferenzen die Anpassung von Ontologie-Alignments unterstützt werden kann.

RG2 Die Graphstruktur der Ontologien wurde bisher bei der Vorhersage der Anpassungen von Alignments an Veränderungen in den Ontologien nur sehr begrenzt einbezogen. Es ist unklar, ob auch die Position eines Knoten im gesamten Graphen für diese Aufgabe genutzt werden kann und wie diese Position repräsentiert werden soll. Dieser Frage wird sich in dieser Arbeit ebenfalls gewidmet.

RG3 Die Anwendung von regelbasierten Ansätzen zur Vorhersage von Veränderungen an Ontologie-Alignments wurde bisher, neben künstlichen Datensätzen, nur auf Biomedizinischen Ontologien untersucht. Weitere Anwendungsdomänen, zum Beispiel Anwendungen aus der Ontologiebasierten Softwareenticklung, wurden bisher nicht betrachtet. Dieser Anwendungsbereich wird in dieser Arbeit untersucht.

RG4 Ein weiterer nicht in der Literatur untersuchter Aspekt ist die Verwendung von Methoden des maschinellen Lernens bei der Anpassung von Ontologie-Alignments an Veränderungen in den Ontologien. Bisher wurde lediglich mit manuellen, statistischen Analysen gearbeitet. In dieser Arbeit wird unter anderem untersucht, ob statistische Eigenschaften des Veränderungsprozesses auch automatisiert genutzt werden können.

RG5 Bei der Betrachtung von Veränderungen in Ontologien wurden bisher in der Literatur ausschließlich manuell festgelegte Features betrachtet. Methoden zum unüberwachten Lernen von Features wurden bisher beim Betrachten von Veränderungen in Ontologien und insbesondere bei der Anpassung von Alignments an Ontologie-Veränderungen nicht verwendet. Dieser Aspekt wird in dieser Arbeit untersucht.

RG6 Beim Erstellen von neuen Alignments und bei der Vervollständigung von Alignments wurden bereits Ansätze zum automatisierten Lernen von Repräsentationen untersucht. Wie sich die Ergebnisse aus diesem Bereich auf die Vorhersage von Änderungen an Alignments übertragen lassen, ist noch unklar. Diese Arbeit betrachtet diese Fragestellung.

Kapitel 4

Problemstellung

Dieses Kapitel beschreibt, wie die Problemstellung, die Anpassung von Ontologie-Alignments an Veränderungen, in dieser Arbeit betrachtet wird. Abschnitt 4.1 stellt einige grundsätzliche Überlegungen vor, wie unterschiedliche Ansätze an das Problem gestaltet werden können und diskutiert die Vor- und Nachteile dieser Gestaltungsmöglichkeiten. Das Ziel der Arbeit wird zusammen mit den Forschungsfragen in Abschnitt 4.2 vorgestellt. Daraufhin wird in Abschnitt 4.3 die Problemstellung präzisiert und es werden zwei Kernprobleme definiert.

4.1 Klassifikation von möglichen Ansätzen

Ansätze zum Anpassen von Ontologie-Alignments an Veränderungen in den verbundenen Ontologien können auf viele Arten klassifiziert werden. In dieser Arbeit wird eine Klassifizierung anhand von Charakteristika vorgenommen, die unterschiedliche Stärken und Schwächen der jeweiligen Ansätze hervorheben. Die hier betrachteten Charakteristika sind: (1) welche Merkmale der Veränderungen und der veränderten Ontologien verwendet werden, (2) wie die Reaktion auf die Veränderungen erfolgt und (3) wie und ob Inferenzen in die Anpassung von Alignments mit einbezogen werden.

(1) **Merkmale der Veränderungen** Eine wichtige Eigenschaft von Ansätzen zur Anpassung von Alignments an Änderungen in Ontologien ist, welche Merkmale von Veränderungen und Ontologien betrachtet werden und wie diese repräsentiert werden. So können zum Beispiel Veränderungsmodelle definiert werden, die festlegen, welche Arten von Veränderungen automatisch verarbeitet werden können. Ein solches Veränderungsmodell wird beispielsweise in [Gro+13] und [Rei+15a] verwendet. Diese Veränderungsmodelle haben den Vorteil, dass sie auf eine spezifische Wissensdomäne angepasst werden können und sich entsprechend gut für spezialisierte Wissensgraphen eignen. Außerdem sind Verände-

rungsmodelle dazu geeignet, von Experten auf Korrektheit überprüft zu werden, da diese Modelle Sachverhalte abbilden, die Domänenexperten aus ihrer Arbeit in der Donäne kennen. Diese Ansätze stoßen allerdings an eine Grenze, wenn sich im Laufe der Zeit die Art, wie sich der Wissensgraph ändert, ebenfalls verändert. Das heißt, dass sich die Veränderungen nicht mehr adäquat in dem bestehenden Veränderungsmodell abbilden lassen. Außerdem lassen sich Veränderungsmodelle aus einer Domäne selten verallgemeinern oder auf andere Domänen übertragen.

Als weiteres Merkmal können Hintergrundinformationen zu den von Veränderungen betroffenen Informationen betrachtet werden. Beispielsweise wird in [Car+18] auf eine Datenbank zu Publikationen zurückgegriffen, um zu erfassen, wie oft ein Element der Ontologie in Publikationen erwähnt wurde. Diese Kennzahl hat einen Einfluss auf die Wahrscheinlichkeit, dass ein solches Element geändert werden muss. Dieser Ansatz ist ebenfalls sehr stark an eine Anwendungsdomäne gebunden, da diese Hintergundinformationen äußerst domänenspezifisch sind.

Außerdem ist es möglich, die Graphstruktur mit einzubeziehen: Merkmale wie der Grad der Graphknoten (die Anzahl an Vorgängern und Nachfolgern) können ebenfalls betrachtet werden [Car+18]. Es ist auch möglich, gelernte Graph-Repräsentationen wie Graph Embeddings zu verwenden (siehe Abschnitt 2.3.1). Sowohl Graph-Embeddings als auch strukturelle Maße von Graphknoten enthalten nur Ausschnitte aus der Strukturinformation, die ein Graph zur Verfügung stellt. Die Graphstruktur kann auch vollständig betrachtet werden. Die Graphstruktur mit einzubeziehen hat den Vorteil, dass die Struktur nur in manchen Fällen an eine konkrete Ontologiesprache gebunden ist und nicht an die konkrete Domäne. Daher ist anzunehmen, dass Ansätze, die die Graphstruktur betrachten, auf verschiedene Domänen anwendbar sind und sich verallgemeinern lassen.

② **Reaktion auf Veränderungen** Die Reaktion auf Veränderungen wird in zwei Kategorien unterteilt: Reaktion auf Veränderungen mithilfe von manuell erstellten, statischen Regeln und Reaktion, die durch maschinelles Lernen festgelegt wird. Auf manuell erstellten Regeln aufbauende oder durch statische Algorithmen vorgegebene Reaktionen haben den Vorteil, dass sie (wie auch Ansätze mit Veränderungsmodellen) von Experten der Domäne nach ihren Erfahrungen festgelegt werden können und sich somit auf Korrektheit überprüfen lassen. Allerdings ist unklar, welche Regeln sich auch auf andere Domänen übertragen lassen. Ein weiterer Vorteil ist, dass sich Entscheidungen eines regelbasierten Algorithmus nachvollziehen lassen. Die Ergebnisse eines solchen Algorithmus sind also erklärbar. Beispiele für Ansätze, die solche Regeln verwenden sind [Gro+13; Rei+15a].

Wird maschinelles Lernen eingesetzt, kann zwar ein trainiertes Modell nicht in anderen Domänen eingesetzt werden, aber es besteht durchaus die Möglichkeit, ein Modell der gleichen Art auf einem neuen Datensatz zu trainieren, ohne dass größere Eingriffe von Domänenexperten notwendig wären. Außerdem ist so eine Anpassung an die Domäne möglich. Dafür kann bei vielen Verfahren des maschinellen Lernens nicht beurteilt werden, nach welchen Kriterien Entscheidungen getroffen werden – eine Überprüfung auf Korrektheit ist also nur schwer durchzuführen und in einigen Fällen sogar unmöglich. Es gibt aber auch Verfahren des maschinellen Lernens, die es ermöglichen, die durch das gelernte Modell getroffenen Entscheidungen zu verstehen. Zum Beispiel existieren Ansätze, die automatisiert gut zu Trainingsdaten passende Regeln lernen [Mei+18]. In dieser Arbeit werden keine solchen Regel-Lerner betrachtet. Entsprechend sind im folgenden immer, wenn Regelbasierten Systeme oder Ansätze diskutiert werden, Verfahren gemeint, bei denen die Regeln manuell beschrieben werden müssen.

③ **Einbeziehung von Inferenzen** Es ist möglich, beim Anpassen von Ontologie-Alignments an Änderungen in den Ontologien zu berücksichtigen, welche Fakten sich durch Inferenz aus dem Wissensgraph ableiten lassen. Allerdings ist noch unklar ob dies tatsächlich notwendig ist. Es existieren bereits Ansätze, die keine Inferenzen berechnen und auf existierenden Datensätzen einsetzbar sind. Inferenzen zu berechnen ist außerdem äußerst aufwendig, dem entsprechend kann es aus Effizienzgründen sinnvoll sein, keine Inferenzen zu berechnen.

Um die Berechnung der Inferenzen auslassen zu können, müssen beim Entwurf eines Algorithmus die Inferenzregeln mit einbezogen werden, um korrekte Ergebnisse zu liefern. Das heißt, Teile der Semantik der Ontologiesprache müssen berücksichtigt werden, damit bei der Anpassung des Alignments keine Inkonsistenzen entstehen oder es zu anderen unerwünschten Änderungen in der Ontologie kommt. Das schränkt den Anwendungsbereich eines Ansatzes (zumindest wenn ein regelbasierter Ansatz gewählt wird) auf genau eine Menge an Inferenzregeln und somit genau eine Ontologiesprache ein. Die Berechnung von Inferenzen führt an dieser Stelle zu einem flexibleren Ansatz, da die Inferenzregeln der Onthologiesprache nicht Bestandteil des eigentlichen Anpassungsalgorithmus sind.

4.2 Ziel der Arbeit/Forschungsfrage

Ziel dieser Arbeit ist es, Methoden zur Unterstützung der Anpassung von Alignments zu entwickeln und zu evaluieren. Dabei soll vor allem untersucht werden, welche Methoden dazu eingesetzt werden können. Im Folgenden werden diese Methoden näher betrachtet, um heraus-

zustellen, welche Vor- und Nachteile sie jeweils mit sich bringen. Aus den zuvor vorgestellten Überlegungen und den in den verwandten Arbeiten vorgestellten Forschungslücken (siehe Abschnitt 3.4) leiten sich die folgenden Forschungsfragen (RQ) ab:

RQ1 Welche Methoden können für die Klassifikation, ob Veränderungen an den Ontologien Veränderungen am Alignment nach sich ziehen, angewendet werden?

Hier sollen insbesondere Verfahren gefunden werden, mit denen der Alignment Adaption Prozess unterstützt werden kann.

RQ2 Welchen Effekt hat das Einbeziehen von Inferenzen in den Prozess zur Anpassung des Alignments an Veränderungen?

Da das Berechnen von Inferenzen sehr aufwändig ist, stellt sich auch die Frage, ob es sinnvoll ist, sie in praktischen Beispielen einzusetzen. Sollte die Einbeziehung von Inferenzen auch in praktischen Anwendungsfällen sinnvoll sein, wird untersucht, welche Einschränkungen dazu in Kauf genommen werden müssen und welche Vorteile sich ergeben.

RQ3 Welche Einschränkungen müssen für Ontologien und Alignments gelten, damit ein regelbasierter Ansatz angewandt werden kann?

RQ4 Ist es möglich die inhärente Graphstruktur von Alignments und Ontologien direkt mit einzubeziehen? Wenn ja, welche Verfahren können dazu verwendet werden?

RQ5 Können moderne Methoden des maschinellen Lernens wie Graph Neural Networks und Knowledge Graph Embeddings zur Anpassung von Alignments an Veränderungen genutzt werden?

RQ6 Können diese Methoden des maschinellen Lernens auch mit Veränderungsmodellen verbunden werden? Welche Vorteile bietet ein solcher Ansatz?

Eine einheitliche Beantwortung dieser Frage setzt einen geeigneten formalen Rahmen voraus. Dieser wird im Folgenden beschrieben.

4.3 Verallgemeinerte Aufgabenstellung

Die in Abschnitt 2.2 vorgestellte Terminologie, die Alignments und Ontologien als Mengen beschreibt, ist nicht geeignet, um die in dieser Arbeit betrachtete Fragestellung hinreichend genau zu fassen, da die verwendeten Ansätze Ontologien als Graphen betrachten. Aus diesem Grund werden Ontologien auch formell als gerichtete Multigraphen mit gelabelten Kanten definiert.

Zwei Ontologien mit zugehörigem Alignment werden dabei gemeinsam betrachtet – es wird also ein Graph verwendet, um sowohl die Ontologien als auch das Alignment abzubilden.

Definition 4.3.1 (Ontologien mit Alignment) *Zwei Ontologien mit einem Alignment werden definiert als ein Gerichteter Multigraph* $G(V, E, s, t, \Lambda)$ *mit*

- $V = V_1 \cup V_2$*, der Menge der Knoten aus beiden Ontologien*
- E*, der Menge der Kanten*
- $s : E \to V$*, der Funktion, die jeder Kante ihren Ursprung zuweist*
- $t : E \to V$*, der Funktion, die jeder Kante ihr Ziel zuweist und*
- $\Lambda : E \to \mathbb{N}$*, der Funktion, die den Kanten ihr jeweiliges Label zuweist*
- $E_a = \{e | (s(e) \in V_i \wedge t(e) \in V_j \wedge i \neq j\}$*, der Menge der Kanten, die das Alignment repräsentieren.*
- $\mathcal{G}$ *bezeichnet die Menge aller solcher Graphen und* $\mathcal{V}$ *die Menge aller Knoten, die in* $\mathcal{G}$ *vorkommen.*

Wenn im weiteren Verlauf dieser Arbeit von einem Graph gesprochen wird, ist immer ein solches Konstrukt aus zwei Ontologien und einem Alignment gemeint, es sei denn, es wird explizit etwas anderes erwähnt. $\mathcal{G}$ ist mit dieser Definition nicht notwendigerweise wohldefiniert. Dies kann umgangen werden, indem V und E auf endliche Mengen (zum Beispiel endliche $V \subsetneq \mathbb{N}$ und $E \subsetneq \mathbb{N}$) eingeschränkt werden.

Im Rahmen dieser Arbeit werden manchmal nur Teile eines Graphen betrachtet, die nicht in einem anderen Graph enthalten sind. Dafür wird die Graph-Differenz wie folgt definiert:

Definition 4.3.2 (Graph-Differenz) *Die Graph-Differenz des Graphen* $G_1(V_1, E_1, s_1, t_1, \Lambda_1)$ *von*
$G_2(V_2, E_2, s_2, t_2, \Lambda_2)$ *,* $G_1 \setminus G_2$ *ist definiert als:*

$$G_1 \setminus G_2 = (s_1(E_1 \setminus E_2) \cup t_1(E_1 \setminus E_2), E_1 \setminus E_2, s_1|_{E_1 \setminus E_2}, t_1|_{E_1 \setminus E_2}, \Lambda_1|_{E_1 \setminus E_2})$$

Außerdem werden Graphen zusammengeführt. Diese Graphen müssen die Vorbedingung erfüllen, dass die Funktionen s, t und Λ für identische Knoten das gleiche Ergebnis zurückliefern. Nur so kann sichergestellt werden, dass jeder Kante nur ein Ursprung, ein Ziel und ein eindeutiges Label zugeordnet werden.

Definition 4.3.3 (Graph-Vereinigung) *Für zwei Graphen* $G_1(V_1, E_1, s_1, t_1, \Lambda_1)$ *und* $G_2(V_2, E_2, s_2, t_2, \Lambda_2)$ *mit* $s_1|_{E_1 \cup E_2} = s_2|_{E_1 \cup E_2}$ *und* $t_1|_{E_1 \cap E_2} = t_2|_{E_1 \cap E_2}$ *und* $\Lambda_1|_{E_1 \cap E_2} = \Lambda_2|_{E_1 \cap E_2}$ *ist die Graph-Vereinigung* $G_1 \cup G_2$ *definiert als:*

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2, s_{1,2}, t_{1,2}, \Lambda_{1,2})$$

- *wobei* $s_{1,2} : E_1 \cup E_2 \to V_1 \cup V_2$ *mit* $s_{1,2}(e) = \begin{cases} s_1(e) & \textit{für } e \in E_1 \\ s_2(e) & \textit{sonst.} \end{cases}$
- *und* $t_{1,2} : E_1 \cup E_2 \to V_1 \cup V_2$ *mit* $t_{1,2}(e) = \begin{cases} t_1(e) & \textit{für } e \in E_1 \\ t_2(e) & \textit{sonst.} \end{cases}$
- *und* $\Lambda_{1,2} : E_1 \cup E_2 \to \mathbb{N}$ *mit* $\Lambda_{1,2}(e) = \begin{cases} \Lambda_1(e) & \textit{für } e \in E_1 \\ \Lambda_2(e) & \textit{sonst.} \end{cases}$

Wird an einem Graph $G(V, E, s, t, \Lambda)$ eine Veränderung vorgenommen, entsteht ein neuer Graph, $G'(V', E', s', t', \Lambda')$. Um festzulegen, welches Element aus dem alten Graph G in dem neuen Graph G' ebenfalls vorhanden ist werden zwei Abbildungsfunktionen I_V und I_E definiert. Wenn aus dem Kontext klar wird, welche der beiden Funktionen verwendet wird, wird die abkürzende Schreibweise I verwendet:

Definition 4.3.4 (Versionsabbildung) *Eine Versionsabbildung zwischen zwei Graphen* $G(V, E, s, t, \Lambda)$ *und* $G'(V', E', s', t', \Lambda')$ *ist durch zwei Funktionen* I_V *und* I_E *definiert mit:*

- $I_V : V \to V' \cup \{\cdot\}$
- $I_V(V_i) \subseteq V_i' \cup \{\cdot\}$
- $I_E : E \to E' \cup \{\cdot\}$
- $I_E(E_i) \subseteq E_i' \cup \{\cdot\}$
- $I_E(E_a) \subseteq E_a' \cup \{\cdot\}$
- $\Lambda' \circ I|_{I^{-1}(E')} = \Lambda|_{I^{-1}(E')}$

wobei $\cdot$ *für Elemente verwendet wird, die nicht abgebildet werden können, um sicherzustellen, dass* I_V *und* I_E *total sind.*

Die Funktionen I_V und I_E ermöglichen die Abbildung von einer Version zur nächsten. V und V' beziehungsweise E und E' müssen nicht disjunkt sein. In dieser Arbeit werden die Graphen so konstruiert, dass die unveränderten Knoten und Kanten jeweils in $V \cap V'$ beziehungsweise $E \cap E'$ liegen.

Für diese Arbeit ist insbesondere die Menge der unveränderten Alignment-Statements interessant.

Definition 4.3.5 (Unveränderte Alignment-Statements) *Die Mengen $\bar{E}_a$ und $\bar{E}'_a$ sind die Mengen der unter I unveränderten Alignment-Statements:*

$$\bar{E}_a := \{e \in E_a | I(e) = e \wedge I(s(e)) = s'(e) \wedge I(t(e)) = t'(e)\}$$

$$\bar{E}'_a := I(\bar{E}_a)$$

Definition 4.3.6 (Von Alignment-Veränderungen betroffene Knoten) *Die Menge M der unter I von Alignment-Veränderungen betroffenen Knoten ist definiert als*

$$M = s(E_a \backslash \bar{E}_a) \cup t(E_a \backslash \bar{E}_a) \cup s'(E'_a \backslash \bar{E}'_a) \cup t'(E'_a \backslash \bar{E}'_a)$$

Mit dieser Notation lässt sich die Problemstellung dieser Arbeit, anhand der die Forschungsfragen RQ1-RQ4 beantwortet werden sollen, genauer fassen:

(1) Wenn ein Graph $G(V, E, s, t, \Lambda)$ sich nach $G'(V', E', s', t', \Lambda')$ verändert, soll die Menge der von Alignment-Veränderungen betroffenen Knoten M vorhergesagt werden, ohne dass E'_a sowie s' und t' vollständig bekannt sind. Zur Evaluation der Ergebnisse sollen verbreitete Klassifikationsmetriken wie Precision und Recall verwendet werden.

Problem 4.3.1 (Von Alignment-Veränderungen betroffenen Knoten finden)

Gegeben *zwei Graphen G und G'*

Gesucht *eine Approximation M^* von M, unter der Nebenbedingung, dass für die Konstruktion von M^* die Information $E'_a, s'|_{E'_a}$ und $t'|_{E'_a}$ nicht verwendet werden darf.*

Güte der Ergebnisse *Precision $\frac{\#(M^* \cap M)}{\#M^*}$ und Recall $\frac{\#(M^* \cap M)}{\#M}$*

(2) Unter den gleichen Voraussetzungen wie in (1) sollen möglichst genaue Approximationen E'^*_a, s'^* und t'^* für E'_a, s' und t' gefunden werden. Zur Beurteilung der Ergebnisse sollen wie in (1) Precision und Recall berechnet werden.

Problem 4.3.2 (Alignment-Approximation Finden) Gegeben *zwei Graphen G und G'*

Gesucht *eine Approximation E'^{*}_{A} von E'_{A}, unter der Nebenbedingung, dass für die Konstruktion von E'^{*}_{A} die Information $E'_{a}, s'|_{E'_{a}}$ und $t'|_{E'_{a}}$ nicht verwendet werden darf.*

Güte der Ergebnisse *Precision* $\frac{\#(E'^{*}_{A} \cap E'_{A})}{\#E'^{*}_{A}}$ *und Recall* $\frac{\#(E'^{*}_{A} \cap E'_{A})}{\#E'_{A}}$

Kapitel 5

Ansatz

In diesem Kapitel wird der Ansatz dieser Arbeit und damit der zentrale Beitrag der vorliegenden Dissertation vorgestellt. Dabei erfolgt auch eine Einordnung des Ansatzes in die bereits in Abschnitt 4.1 skizzierten Kategorien von Verfahren zur automatisierten Anpassung von Alignments an Veränderungen in Ontologien, die durch diese Alignments vernetzt sind.

Zunächst wird in Abschnitt 5.1 vorgestellt, wie *Inferenzen* in die automatisierte Reaktion auf Änderungen in Ontologien, die durch Alignments vernetzt sind, einbezogen werden können. Dazu wird ein Ansatz beschrieben, der Inferenzen einbezieht und *regelbasiert* auf Veränderungen reagiert. Die Inferenzen werden vor und nach einer Änderung berechnet. Auf den Unterschieden dieser Inferenzen werden daraufhin Regeln angewandt. Diese Regeln legen fest, wie das Alignment angepasst wird.

Ein weiterer Ansatz, welcher in Abschnitt 5.2 vorgestellt wird, untersucht inwiefern maschinelles Lernen zur Anpassung von Alignments an Veränderungen in Ontologien eingesetzt werden kann. Dazu wird eine für die Anwendung auf das Mapping Adaption Problem neuartige Repräsentationsmethode zur Darstellung der vernetzten Ontologien verwendet, sogenannte *Knowledge Graph Embeddings*, welche in Abschnitt 2.3.1 vorgestellt wurden. Mit dieser Repräsentationsform wird mit Verfahren des maschinellen Lernens automatisiert vorhergesagt, welche Alignment-Statements nach einer Änderung der Ontologien angepasst werden müssen.

Da der Ansatz zur Einbeziehung von Knowledge Graph Embeddings ausschließlich die Struktur der Ontologie und nicht die Art der Veränderungen betrachtet, wird noch ein weiterer Ansatz vorgestellt, der diesen Aspekt mit einbezieht. Dazu wird ein *Veränderungsmodell* verwendet, dass die Art der Veränderungen kategorisiert. Die Klassen des Veränderungsmodells werden mit der durch die Embeddings erzeugten strukturellen Repräsentationen kombiniert und dann mit Verfahren des maschinellen Lernens verarbeitet. Dieser Ansatz wird in Abschnitt 5.3 vorgestellt.

Schließlich wird in Abschnitt 5.4 ein Ansatz vorgestellt, der untersucht, wie auch ohne

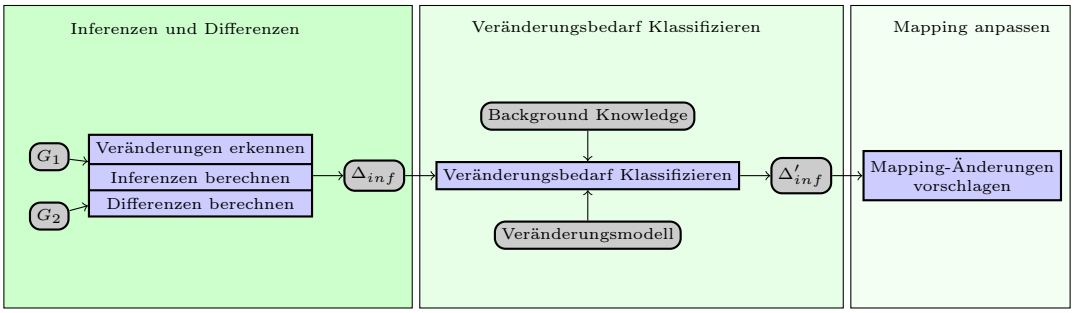

Abbildung 5.1: Überblick Ansatz zur Einbeziehung von Inferenzen

einen Zwischenschritt zur Erstellung der Repräsenation automatisiert auf Veränderungen reagiert werden kann. Dazu wird mit speziell auf Graphen ausgerichteten Klassifikationsverfahren, sogenannten *Graph Neural Networks*, gearbeitet.

5.1 Einbeziehung von Inferenzen

Bei der Reaktion auf Veränderungen in vernetzten Ontologien wurde in der Forschung bisher die Semantik von Veränderungen kaum betrachtet [SE13]. Ontologien bieten die Möglichkeit, die Semantik von Wissensrepräsentationen über Inferenzen abzubilden. Dieses Vorgehen wurde bisher im Kontext der Anpassung von Alignments an Ontologie-Veränderungen noch nicht behandelt. Zwar stellt [Rei+15b] sogenannte *Semantic Change Patterns* vor (siehe Abschnitt 3 für eine nähere Diskussion), aber dieser Ansatz ist auf einen festen Satz von spezifischen Regeln an die jeweilige Ontologiesprache angepasst. Wird die Ontologiesprache geändert, müssen neue Regeln erstellt werden. Eine flexiblere Lösung bezieht Inferenzen direkter ein. Dieser Abschnitt stellt einen solchen Ansatz [Jur16] vor.

Ein Überblick über diesen Ansatz ist in Abbildung 5.1 dargestellt. Im ersten Schritt werden Inferenzen berechnet. Um die Veränderungen der Inferenzen zu betrachten, werden die Inferenzen vor und nach einer Veränderung berechnet. Diese Mengen von Inferenzen können dann miteinander verglichen werden, was zu einer Menge von hinzugekommenen und entfernten Inferenzen führt, die als Δ_{inf} bezeichnet wird. Aus dieser Menge werden jene Elemente ausgewählt, für die das Alignment anzupassen ist. Bei der Anpassung können sowohl Hintergrundwissen als auch Veränderungsmodelle und Regeln hinzugezogen werden. In dieser Arbeit wird allerdings die Einbindung von Hintergrundwissen nicht betrachtet und hier nur der Vollständigkeit halber aufgeführt. Auf die Elemente, bei denen ein Reaktionsbedarf festgestellt wurde (Δ'_{inf}) werden manuell definierte Regeln zur Anpassung des Alignments angewandt. Solche Regeln definieren zum Beispiel, dass bestimmte Statements aus dem Alignment entfernt oder andere Statements hinzugefügt werden.

5.1.1 Berechnen der Inferenzen

Um die Inferenzen zu berechnen wird das System aus Alignment und Ontologien wie in Abschnitt 4.3 beschrieben als Graph betrachtet. Eingabe für die Berechnung der Inferenzen sind die vollständigen Ontologien mit dem alten Alignment, sowohl vor als auch nach den Änderungen an den Ontologien. Es werden also zwei Graphen aus Ontologien und Alignments betrachtet: Einer vor und einer nach der Änderung. Formal wird das Hinzufügen von Inferenzen als das Erweitern eines Graphen nach festgelegten Regeln betrachtet. Eine Graph-Erweiterung wird wie folgt gefasst:

Definition 5.1.1 (Graph-Erweiterung) *Wenn $G(V, E, s, t, \Lambda)$ einen Graph aus zwei Ontologien und einem Alignment dieser Ontologien bezeichnet, dann bezeichnet $G^*(V^*, E^*, s^*, t^*, \Lambda^*)$ eine Graph-Erweiterung von G mit*

- $V \subseteq V^*$
- $E \subseteq E^*$
- $s^*|_E = s \wedge t^*|_E = t$
- $\Lambda^*|_V = \Lambda$

Die Regeln, nach denen die Inferenzen selbst berechnet werden, sind an dieser Stelle nicht formalisiert, um möglichst unabhängig von den jeweiligen Regeln für Inferenz-Berechnung und somit von der konkreten Ontologiesprache zu sein. Das Berechnen und Hinzufügen von Inferenzen stellt dann eine Graph-Erweiterung im Sinne der Definition 5.1.1 dar. Wenn im folgenden das Symbol einer Graph-Erweiterung verwendet wird, ist damit der um Inferenzen erweiterte Graph gemeint, es sei denn, es wird explizit anders angegeben.

Da für Ontologien signifikanter Größe die Berechnung aller Inferenzen nicht praktikabel ist (beispielsweise ist die Berechnung von OWL-Inferenzen für einige OWL-Dialekte unentscheidbar und für viele Reasoning-Tasks NP-vollständig oder härter [Mot+12]), kann in praktischen Anwendungsfällen nur ein Teil der Inferenzen berechnet werden. In dem hier vorgestellten Ansatz wird dieses Problem umgangen, indem Inferenzen nur für Ausschnitte der Ontologie berechnet werden. Die Ausschnitte der Ontologie werden durch die *lokale Umgebung der Veränderungen* gebildet: Alle Knoten, die in n Schritten von einem veränderten Knoten erreichbar sind, zählen zu der Nachbarschaft dieses Knoten. Der Wert von n muss dabei experimentell, je nach Anwendungsfall, bestimmt werden.

Definition 5.1.2 (In einer festen Anzahl von Schritten erreichbare Knoten) *Wenn $G(V, E, s, t, \Lambda)$ einen Graphen bezeichnet, dann ist die Funktion der von einem Knoten in einer festen Anzahl n von Schritten erreichbaren Knoten $\rho : (V, \mathbb{N}\backslash 0) \to \mathcal{P}(V)$ definiert als*

$$\rho(v, n) = \begin{cases} \{v_r | \exists e \in E : (s(e) = v \wedge t(e) = v_r) \vee (s(e) = v_r \wedge t(e) = v)\} & \text{für } n = 1 \\ \bigcup\limits_{v_r \in \rho(v, n-1)} \rho(v_r, 1) & \text{sonst.} \end{cases}$$

Diese Definition beachtet die Richtung der Kanten im Graphen nicht. Dies ist im Fall von Wissensgraphen durchaus sinnvoll: in welcher Richtung eine Beziehung gelesen wird, kann beliebig geändert werden. Zum Beispiel kann eine Vorfahren-Beziehung einfach durch eine Nachfahren-Beziehung abgebildet werden, ohne dass sich die Semantik der Darstellung ändert.

Definition 5.1.3 (Nachbarschaft von Knoten) *Wenn $G = (V, E, s, t, \Lambda)$ einen Graph bezeichnet, dann bezeichnet $G_{c,n}(V_{c,n}, E_{c,n}, s_{c,n}, t_{c,n}, \Lambda_{c,n})$ den Teilgraph von G, der alle Knoten aus $\rho(c, n)$ und alle Kanten aus $e \in E$ mit $s(e) \in \rho(c, n) \vee t(e) \in \rho(c, n)$ enthält.*

Dieses Konzept wird genutzt, um Inferenzen nur für relevante Teile des Graphen zu berechnen: Die Nachbarschaft bildet einen Ausschnitt aus der Ontologie, der je nach Wahl von n wesentlich kleiner sein kann als die gesamte Ontologie. Dementsprechend können auf diesem Ausschnitt der Ontologie deutlich schneller Inferenzen berechnet werden.

Um die Auswirkungen von Veränderungen der Ontologie beurteilen zu können, wird dieses Nachbarschaftskonzept auf veränderte Knoten angewandt. Zu jeder Veränderung werden die Inferenzen vor und nach der Veränderung berechnet. Dazu wird die partielle Inferenz eines Graphen betrachtet:

Definition 5.1.4 (Partielle Inferenz) *Wenn $G = (V, E, s, t, \Lambda)$ ein Graph ist und $C \subseteq V$ eine Menge von Knoten aus V, dann ist*

$$\Phi_{C,n}(G) = \bigcup_{c \in C} G^*_{c,n}$$

der Graph, der aus allen zu $c \in C$ gehörenden Teilgraphen abgeleitet werden kann.

Wenn C die Menge aller Knoten ist, die verändert wurden, ist es möglich, die hinzugekommenen und entfernten Inferenzen zu berechnen:

Definition 5.1.5 (Hinzugekommene und entfernte Inferenzen) *$\Phi_{C,n}(G)$ und $\Phi_{C,n}(G')$ bezeichnen die durch Inferenz vervollständigten Graphen vor und nach einer Veränderung. Der Graph*

$$\Delta_{add} = \Phi_{C,n}(G') \backslash \Phi_{C,n}(G)$$

bezeichnet den Graph der hinzugekommenen und der Graph

$$\Delta_{del} = \Phi_{C,n}(G) \backslash \Phi_{C,n}(G')$$

den Graph der entfernten Inferenzen.

Ein Graph, der aus hinzugekommenen oder entfernten Inferenzen besteht wird im weiteren Verlauf der Arbeit mit G_Δ bezeichnet.

Wenn über die einzelnen Elemente der Graphen mit den hinzugekommenen und entfernten Inferenzen gesprochen wird, werden $G_{add}(V_{add}, E_{add}, s_{add}, t_{add}, \Lambda_{add})$ und $G_{del}(V_{del}, E_{del}, s_{del}, t_{del}, \Lambda_{del})$ als ausführlichere Schreibweise für Δ_{add} und Δ_{del} verwendet.

Durch die Anwendung der hinzugekommenen und entfernten Inferenzen können die Veränderungen der Ontologien und teile ihrer Semantik als Graph betrachtet werden. Dies ist die Grundlage für die Regeln, die definieren, wie auf Veränderungen reagiert werden soll.

5.1.2 Regeln

Auf die beschriebenen Graphen von hinzugekommenen und entfernten Inferenzen werden dann Regeln angewendet. Im einfachsten Fall können Regeln bestimmen, wie auf das Entfernen eines Tripels reagiert wird. Diese Regeln können zum Beispiel so gestaltet sein, dass nach konkreten URIs in den Graphen Δ_{add} und Δ_{del} gesucht wird. So könnte zum Beispiel eine Regel festlegen, dass in einer entfernten Kante e die URI von $t(e)$ bestimmt, ob eine Aktion notwendig ist und je nach der URI von $s(e)$ wird unterschiedlich reagiert. Die Regeln sind in zwei Schritte aufgeteilt:

(1) Zunächst muss festgestellt werden, ob eine Änderung notwendig ist. Dazu wird auf jede Kante aus diesen Graphen Δ_{add} und Δ_{del} eine Funktion zur Bestimmung des Änderungsbedarfs angewandt:

Definition 5.1.6 (Bestimmung des Änderungsbedarfs) *Eine Funktion zur Bestimmung des Änderungsbedarfs γ für eine gegebene Kantenmenge E ist eine Funktion*

$$\gamma : E \to \{\boxed{a}, \boxed{n}\}.$$

$\boxed{a}$ markiert dabei die Kanten, zu denen eine Alignment-Änderung erfolgen muss und $\boxed{n}$ markiert die Kanten, zu denen keine weitere Aktion erforderlich ist. Die Funktion zur Bestimmung des Änderungsbedarfs wird für Δ_{add} und Δ_{del} getrennt definiert. Die Funktion für Kanten aus Δ_{add} wird als γ_{add} und die Funktion für Kanten aus Δ_{del} als γ_{del} bezeichnet.

(2) Die Reaktion auf Veränderungen wird ebenfalls durch eine Funktion definiert. In dieser Arbeit werden auf die durch γ mit $\boxed{a}$ markierten Kanten zwei Funktionen angewandt: eine Funktion zur Berechnung der Ergänzungen des Alignments und eine Funktion zur Berechnung der aus dem Alignment zu entfernenden Kanten.

Definition 5.1.7 (Berechnung der Graph-Ergänzungen) *Eine Funktion α^+ zur Berechnung von Graph-Ergänzungen für einen gegeben Graph $G(V, E, s, t, \Lambda)$ und einen Graph der entweder hinzugekommene (Δ_{add}) oder entfernten Inferenzen (Δ_{del}) darstellt $G_\Delta(V_\Delta, E_\Delta, s_\Delta, t_\Delta, \Lambda_\Delta)$ ist eine Funktion*

$$\alpha^+ : E_\Delta \to \mathcal{G}$$

und für alle $e \in E_\Delta$ mit $\alpha^+(e) = G^+$ und $G_e^+(V_e^+, E_e^+, s_e^+, t_e^+, \Lambda_e^+)$ gilt:

- *$E \cap E_e^+ = \emptyset$ (Es werden nur neue Kanten hinzugefügt)*
- *$\forall e^+ \in E_e^+ : (s_e^+(e^+) \in V_i) \wedge (t_e^+(e^+) \in V_j) \Rightarrow i \neq j$ (Es werden nur Alignment-Kanten hinzugefügt)*

Definition 5.1.8 (Berechnung der Graph-Entfernungen) *Eine Funktion α^- zur Berechnung von Graph-Entfernungen für einen gegeben Graph $G(V, E, s, t, \Lambda)$, und einen Graph der hinzugekommenen (Δ_{add}) oder entfernten Inferenzen (Δ_{del}) $G_\Delta(V_\Delta, E_\Delta, s_\Delta, t_\Delta, \Lambda_\Delta)$ ist eine Funktion*

$$\alpha^- : E_\Delta \to \mathcal{G}$$

und für alle $e \in E_\Delta$ mit $\alpha^-(e) = G^-$ und $G_e^-(V_e^-, E_e^-, s_e^-, t_e^-, \Lambda_e^-)$ gilt:

- *$E_e^- \subseteq E_A$. (Es werden nur aus dem Alignment Kanten entfernt)*

Auch diese Funktionen müssen für Δ_{add} und Δ_{del} getrennt definiert werden. Eine Funktion, die auf Kanten aus Δ_{add} angewandt wird, wird mit α_{add} bezeichnet. Eine Funktion, die auf Kanten aus Δ_{del} angewandt wird, wird mit α_{del} bezeichnet.

Mit den Funktionen zur Berechnung des Änderungsbedarfs und der Funktion zur Berechnung der Änderungen lässt sich ein Algorithmus für die Anwendung von Regeln beschreiben,

Algorithmus 1 Regelbasierter Algorithmus zur Reaktion auf Veränderungen

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Graph $\tilde{G}(\tilde{V}, \tilde{E}, \tilde{s}, \tilde{t}, \tilde{\Lambda})$
Input: Funktion $I_V : V \to \tilde{V}$
Input: Funktion $I_E : E \to \tilde{E}$
procedure BERECHNUNG DER ALIGNMENT-ÄNDERUNGEN
  $C \leftarrow$ Veränderte Knoten C von G nach $\tilde{G}$ $C \subseteq V$
  berechne $\Phi_{C,n}(G)$
  berechne Δ_{add}
  berechne Δ_{del}
  $G_{add}(V_{add}, E_{add}, s_{add}, t_{add}, \Lambda_{add}) \leftarrow \Delta_{add}$
  $G^+(V^+, E^+, s^+, t^+, \Lambda^+)$ ist leerer Graph
  $G^-(V^-, E^-, s^-, t^-, \Lambda^-)$ ist leerer Graph
  for all $e \in E_{add}$ **do**
    if $\gamma_{add}(e) = \boxed{a}$ **then**
      $G^+ \leftarrow G^+ \cup \alpha^+_{add}(e)$
      $G^- \leftarrow G^- \cup \alpha^-_{add}(e)$
    end if
  end for
  $G_{del}(V_{del}, E_{del}, s_{del}, t_{del}, \Lambda) \leftarrow \Delta_{del}$
  for all $e \in E_{del}$ **do**
    if $\gamma_{del}(e) = \boxed{a}$ **then**
      $G^+ \leftarrow G^+ \cup \alpha^+_{del}(e)$
      $G^- \leftarrow G^- \cup \alpha^-_{del}(e)$
    end if
  end for
  return $(\tilde{G} \cup G^+) \setminus G^-$
end procedure

der Inferenzen einbezieht. Dieser Algorithmus ist in Algorithmus 1 abgebildet. Eingabe sind zwei Graphen: Der Graph $G(V, E, s, t, \Lambda)$ und der Graph $\tilde{G}(\tilde{V}, \tilde{E}, \tilde{s}, \tilde{t}, \tilde{\Lambda})$ mit $\tilde{V} = V', \tilde{E} = (E' \setminus E'_A) \cup (E_A \cap E'), \tilde{s} = s'|_{\tilde{E}}, \tilde{t} = t'|_{\tilde{E}}$ und $\tilde{\Lambda} = \Lambda'|_{\tilde{E}}$, welcher die veränderte Ontologie mit noch nicht angepasstem Alignment beschreibt. Zunächst werden aus den Veränderungen Δ_{add} und Δ_{del} berechnet. Für alle Kanten $e \in E_{add}$ im Graphen Δ_{add} wird die Funktion zur Berechnung des Änderungsbedarfs γ_{add} angewandt. Falls γ_{add} einen Veränderungsbedarf feststellt, wird durch die Funktionen zur Berechnung der Änderungen α^+_{add} und α^-_{add} ein Graph mit hinzuzufügenden Kanten und ein Graph mit zu entfernenden Kanten erzeugt. Analog wird für die Kanten im durch Δ_{del} beschriebenen Graphen mit den Funktionen $\gamma_{del}, \alpha^+_{del}$ und α^-_{del} vorgegangen. So erhält man zwei Graphen, G^+ und G^-, die alle hinzuzufügenden und zu entfernenden Kanten enthalten. Der ursprüngliche Graph G, aus dem die Kanten aus G^- entfernt wurden und zu dem die Kanten aus G^+ hinzugefügt wurden, ist das Ergebnis der Berechnung. Neben der hier vorgestellten Beschreibung von Regeln sind viele weitere Alternativen für Regeln und Regelsprachen denkbar, insbesondere auch mit unterschiedlichen Ausdrucksmächtigkeiten. Allerdings sind Regelsprachen nicht im Fokus dieser Arbeit und werden dementsprechend nicht weiter betrachtet.

5.1.3 Anwendbarkeit des Ansatzes

Die Überlegungen zum Ansatz zur Einbeziehung von Veränderungen sind sehr allgemein gehalten, um sicherzustellen, dass eine Anwendbarkeit auf möglichst viele verschiedene Anwendungsgebiete gegeben ist. Um zu zeigen, dass der Ansatz sinnvoll angewandt werden kann, wird im folgenden eine Menge von Einschränkungen vorgestellt. Die Einschränkungen zeigen ein Szenario, in dem der Ansatz in jedem Fall funktioniert und immer eine Lösung berechnet werden kann. Die Einschränkungen sind sehr eng gefasst, um möglichst einfach zu demonstrieren, dass der Ansatz berechenbar ist. Es sind allerdings auch deutlich weiter gefasste Einschränkungen denkbar, mit denen der Ansatz immer noch angewandt werden kann.

Die Einschränkungen werden in drei Gruppen von Einschränkungen aufgeteilt: (1) Vorbedingungen, die sich zwingend aus dem Ansatz ergeben, (2) Einschränkungen, die die Laufzeit des Algorithmus beschränken oder den Umgang mit Datenstrukturen einfacher gestalten und (3) Einschränkungen, die Berechnungsvorschriften für Funktionen festlegen, die im vorherigen Abschnitt in erster Linie über Einschränkungen definiert wurden.

(1) Die erste Gruppe von Einschränkungen ergibt sich in erster Linie aus den Eingaben von Algorithmus 1:

Vorbedingung 5.1.1 (Eingaben) *Die Graphen G, $\tilde{G}$ und die Funktionen I_V und I_E sind gegeben.*

(2) In der Zweiten Gruppe von Einschränkungen wird festgelegt, dass ausschließlich Subclass-Reasoning verwendet wird. Subclass-Reasoning benötigt deutlich weniger Rechenzeit als ausgefeiltere Reasoning-Verfahren. Außerdem wird festgelegt, dass die verschiedenen Graph-Versionen abegesehen von den Änderungen beim Versionssprung identisch sind:

Vorbedingung 5.1.2 (Identische Graphen außerhalb von Veränderungen) *Der Graph $G \backslash \tilde{G}$ besteht ausschließlich aus aus G entfernten Elementen und $\tilde{G} \backslash G$ besteht ausschließlich aus zu G hinzugefügten Elementen.*

I_V und I_E sind also jeweils Funktionen, die Knoten beziehungsweise Kanten auf sich selbst abbilden. Das bedeutet auch, dass G und $\tilde{G}$ abgesehen von Graph-Veränderungen die gleiche Knoten- und Kantenmenge haben. Weiterhin wird festgelegt, dass die einzigen Veränderungen das Löschen und Hinzufügen von Knoten und Kanten sind:

Vorbedingung 5.1.3 (Typen von Graph-Änderungen) *Alle Änderungen des Graphen G, die zu $\tilde{G}$ bestehen aus dem Entfernen oder Hinzufügen von Knoten und Kanten. Wird ein Knoten c entfernt, werden außerdem alle Kanten e mit $s(e) = c \wedge t(e) = c$ entfernt. Änderungen am Alignemnt sind nur als Konsequenz aus der Entfernung eines Knoten zulässig.*

Außerdem wird festgelegt, dass alle Änderungen sich auf eine Ontologie beziehen:

Vorbedingung 5.1.4 (Einschränkung der veränderbaren Ontologie) *Es werden nur Knoten aus V_1 entfernt oder hinzugefügt.*

Dementsprechend werden auch nur Kanten mit $s(e) \in V_1 \vee t(e) \in V_1$ entfernt.

Zusätzlich wird festgelegt, in welcher Reihenfolge die Alignments notiert werden. Der Ursprung der Alignment-Kante ist immer in V_1.

Vorbedingung 5.1.5 (Notation von Alignment-Kanten) *Für alle $e \in E_A$ gilt $s(e) \in V_1$.*

(3) In der dritten Gruppe von Einschränkugen müssen Berechnungsvorschriften für die Funktionen $\gamma_{add}, \gamma_{del}, \alpha^+_{add}, \alpha^-_{add}, \alpha^+_{del}$ und α^-_{del} angegeben werden. Die Funktion zur Bestimmung des Änderungsbedarfs bei entfernten Inferenzen γ_{del} ist die einfachste der Funktionen und wird in Algorithmus 2 dargestellt: Im Fall von γ_{del} wird berechnet, ob die entfernten Kanten Alignment-Statements sind. In dem Falle wird $\boxed{a}$ zurückgegeben, ansonsten $\boxed{n}$.

Die Funktion α^-_{del}, die festlegt, welche Statements als Reaktion auf entfernte Inferenzen aus dem Graph entfernt werden müssen ist sogar noch einfacher: diese Funktion gibt immer einen leeren Graphen zurück. Für die anderen Funktionen wird mehr Wissen über die Knoten im

Algorithmus 2 Algorithmus zur Berechnung der Funktion γ_{del}

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Kante e
procedure γ_{del}
  if $s(e) \in V_1 \wedge t(e) \in V_2$ **then**
    return $\boxed{a}$
  else
    return $\boxed{n}$
  end if
end procedure

Graphen benötigt. Dazu wird eine Datenbasis mit Hintergrundwissen über die im Graph enthaltenen Knoten festgelegt. Diese Datenbasis wird so gestaltet, dass für alle Paare von Knoten aus unterschiedlichen Ontologien, die durch ein Alignment-Statement verbunden werden können, ein Score festgelegt wird. Dieser Score beschreibt, wie gut diese Knoten geeignet sind, durch ein Alignemnt-Statement verbunden zu werden und dementsprechend, wie gut ein Knoten in einem Alignment ersetzt werden kann. Formal wird diese Struktur wie folgt gefasst:

Definition 5.1.9 (Ersetzbarkeit von Alignment-Knoten) *Die Knotenähnlichkeitsfunktion $\rho : V_1 \times V_2 \rightarrow [0, 1]$ legt fest, wie sehr zwei Knoten geeignet sind, über ein Alignment verbunden zu sein.*

Mit diesem Hintergrundwissen lässt sich die Funktion α^{+}_{del} zum Hinzufügen von Statements als Reaktion auf entfernte Inferenzen wie folgt gestalten (Algorithmus 3): Für die mit einer entfernten Klasse verbundene Klasse $t(e)$ wird dann ein neuer Graph erzeugt, der ein Alignment-Statement zwischen c und $t(e)$ enthält und als Ergebnis zurückgegeben. Da Alignments üblicherweise nur Ontologieklassen betrachten, werden an dieser Stelle ebenfalls nur die Knoten betrachtet, die Klassen repräsentieren.

Die Funktion zur Bestimmung des Änderungsbedarfs bei hinzugekommenen Inferenzen (γ_{add}) verwendet ebenfalls dieses Hintergrundwissen. Die Berechnung von γ_{add} ist in Algorithmus 4 dargestellt: Wenn eine neue Klasse hinzugefügt wird, wird mithilfe des Hintergrundwissens überprüft, ob diese Klasse ein besserer Kandidat für ein Alignment-Statement ist als bestehende Teile des Alignments. Ist dies der Fall, ist ein Änderungsbedarf gegeben.

Ähnlich werden die Funktionen α^{+}_{add} und α^{-}_{add} gestaltet. Der Algorithmus für $\alpha_{add}+$ ist in Algorithmus 5 dargestellt. Für eine hinzugefügte Klasse wird untersucht, bei welchen Alignment-Statements die hinzugefügte Klasse eine andere Klasse ersetzen kann. Dann werden entsprechende Ergänzungen generiert.

α^{-}_{add} (siehe Algorithmus 6) sucht ebenfalls nach Kanten, in denen eine Klasse durch die neue Klasse ersetzt werden kann. Diese Kanten werden dann zu einem Graphen der Graph-

Algorithmus 3 Algorithmus zur Berechnung der Funktion α^{+}_{del}

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Kante e
procedure α^{+}_{del}
  Finde c mit größtem $\rho(c, t(e))$
  $V^r_1 \leftarrow c$
  $V^r_2 \leftarrow t(e)$
  $E^r \leftarrow$neue Kante e^r
  $s^r \leftarrow$Funktion mit $s^r(e^r) = c$
  $t^r \leftarrow$Funktion mit $t^r(e^r) = t(e)$
  $\Lambda^r \leftarrow$ Funktion mit $\Lambda^r(e^r) = \Lambda(e)$
  Return $G^r(V^r_1 \cup V^r_2, E^r, s^r, t^r, \Lambda^r)$
end procedure

Algorithmus 4 Algorithmus zur Berechnung der Funktion γ_{add}

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Kante e
procedure γ_{add}
  if e definiert neue Klasse c **then**
    for all $e_a \in E_A$ **do**
      if $\rho(c, t(e_a) > \rho(s(e_a), t(e_a))$ **then**
        Return $\boxed{a}$
      end if
    end for
  else
    Return $\boxed{n}$
  end if
end procedure

Entfernungen hinzugefügt. Die Eingabe e ist dabei das Statement, dass eine Klasse Definiert. In einer mit OWL formulierten Ontologie würde e einen Knoten über einen Kantentyp, der die Unterklassenbeziehung repräsentiert mit der Oberklasse aller Klassen verbinden.

Algorithmus 5 Algorithmus zur Berechnung der Funktion α^+_{add}

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Kante e
procedure α^+_{add}
  $c \leftarrow$ von e definierte Klasse
  $E^+_A \leftarrow \{\}$
  for all $e_a \in E_A$ **do**
    if $\rho(c, t(e_a) > \rho(s(e_a), t(e_a))$ **then**
      e^+_a ist eine neue Kante
      $E^+_A \leftarrow E^+_A \cup \{e^+_a\}$
      $s^+ \leftarrow s^+ \cup s^+(e^+_a) = c$
      $t^+ \leftarrow t^+ \cup t^+(e^+_a) = t(e_a)$
    end if
  end for
  Return Graph $G^+(V \cup \{c\}, E^+_A, s^+, t^+, \Lambda)$
end procedure

Diese Einschränkungen sind derart gestaltet, dass der beschriebene Ansatz auf jeden Fall funktioniert. Das bedeutet allerdings nicht, dass nicht auch weichere Einschränkungen denkbar sind. Viele dieser Einschränkungen wurden so gewählt, dass eine Umsetzung möglichst einfach ist. Prinzipiell könnten viele dieser Einschränkungen auch weiter oder anders gefasst werden, zum Beispiel könnten andere Reasoning-Verfahren oder andere Sprachen zur Beschreibung von Regeln verwendet werden. Die erste Einschränkung ist die einzige, die sich zwingend aus dem Ansatz ergibt.

Ein Beispiel für einen Anwendungsbereich, in dem diese Einschränkungen gelten, wäre die Anwendung in Interweaving Systems. Interweaving Systems sind Systeme, die sich gegenseitig durch eine gemeinsame Umgebung zur Laufzeit beeinflussen können, worauf beim Entwurf der Systeme allerdings nicht eingegangen wurde [Tom+16]. Solche Systeme arbeiten normalerweise unter Bedingungen, die weiche oder harte Echtzeiteinschränkungen erfordern. Das wichtigste Ziel der Entwicklung ist also, dass die Systeme diese Echtzeitanforderungen erfüllen, die Optimierung von Effizienz der Systeme bei der Erfüllung domänenspezifischer Aufgaben wird in der Regel als weniger wichtig angesehen. Die explizite Zusammenarbeit zwischen Interweaving Systems kann diese Aspekte potenziell verbessern, ohne die Echtzeit-/Sicherheitskriterien zu berühren. Ob und wie sich eine gemeinsame Sicht auf die Umgebungen von Interweaving Systems auf die Zusammenarbeit und Leistung in Bezug auf domänenspezifische Ziele auswirkt,

Algorithmus 6 Algorithmus zur Berechnung der Funktion α^{-}_{add}

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Kante e
procedure α^{-}_{add}
  $c \leftarrow$ von e definierte Klasse
  $E^{-}_{A} \leftarrow \{\}$
  $V^{-} \leftarrow \{\}$
  for all $e_a \in E_A$ **do**
    if $\rho(c, t(e_a) > \rho(s(e_a), t(e_a))$ **then**
      $V^{-} \leftarrow V^{-} \cup \{s(e_a), t(e_a)\}$
      $E^{-}_{A} \leftarrow E^{-}_{A} \cup \{e_a\}$
      $s^{-} \leftarrow s^{-} \cup s^{-}(e_a) = s(e_a)$
      $t^{-} \leftarrow t^{-} \cup t^{-}(e_a) = t(e_a)$
    end if
  end for
  Return Graph $G^{-}(V^{-}, E^{-}_{A}, s^{-}, t^{-}, \Lambda)$
end procedure

ist eine offene Frage. Ein erster Schritt zur Lösung dieses Problems besteht darin, einen Ansatz zum Erzeugen einer gemeinsamen Sicht zu entwerfen und zu bewerten. Da die typische Umgebung von Interweaving Systems anfällig für Veränderungen ist und der Ausfall von Systemen in diesem Kontext immer mit einbezogen werden sollte, müssen geeignete Ansätze adäquat auf Veränderungen reagieren können. Im weiteren Verlauf der Arbeit wird im Rahmen der Evaluation in Abschnitt 7.2 ein Ontologiebasierter Ansatz vorgestellt, der diese Einschränkungen erfüllt.

Es gibt allerdings auch einige Faktoren, die dazu führen können, dass der Ansatz bereits aus konzeptionellen Gründen nicht verwendet werden kann. Der Kern dieser Faktoren betrifft die in dem Ansatz zur Einbeziehung von Inferenzen verwendeten Regeln, die händisch erstellt werden müssen. Wenn zum Beispiel die Strukturen in den Inferenzen, aus denen die Regeln abgeleitet werden, für den Menschen nur schwer zu erkennen sind, wird es schwer fallen, diese Regeln zu erstellen. Auch wenn Strukturen einfach zu erkennen sind, muss ein Experte für die jeweilige Domäne der Ontologien viel Zeit investieren und über das notwendige technische Know-How verfügen, um die Regeln zu erstellen. Das führt zu einem hohen Zeitaufwand und somit zu hohen Kosten, um ein solches System aufzusetzen. Außerdem gibt es keine etablierte Systematik, um für Ontologieveränderungen Regeln zu erstellen.

Für große, von Communities gepflegte Ontologien wie DBPedia [Aue+07] oder Wikidata [VK14] ist es äußerst schwierig, Regeln manuell festzulegen. Solche Ontologien sind in der Regel inkonsistent und enthalten aufgrund der vielen Bearbeiter häufig Fehler. Fehler in einer Ontologie führen zwangsweise auch zu Inferenzen, die Fehler enthalten. Es kommt also zu einer

Vervielfältigung der Fehler. Dadurch können Inferenzen nicht mehr ohne weiteres zur Reaktion auf Veränderungen herangezogen werden.

Auch wenn eine umfangreiche, durch eine Community gepflegte Ontologie keine Fehler enthalten würde, ist das Erfassen aller notwendigen Regeln aufgrund der Größe der Ontologie entweder mit sehr viel Aufwand verbunden oder in der Praxis nicht möglich. Dementsprechend ist der vorgestellte regelbasierte Ansatz für diese Ontologien vermutlich nicht praktikabel. Damit der Ansatz gewinnbringend eingesetzt werden kann, müssten die Ontologien eine begrenzte Größe haben und immer widerspruchsfrei sein. Außerdem müssen sich alle Veränderungen der Ontologie durch einen festen und überschaubaren Satz an Regeln fassen lassen, um den regelbasierten Ansatz verwenden zu können.

Eine andere Möglichkeit wäre, die Regeln mithilfe von maschinellem Lernen zu erzeugen. Es existieren bereits Verfahren zum Lernen von Regeln auf Graphdaten, unter anderem vorgestellt in [Mei+18]. Dazu müsste aber eine Systematik gefunden werden, wie solche Regeln für die Anpassung von Alignments an Veränderungen gesucht werden können. Eine solche Systematik ist bisher nicht bekannt. Im folgenden werden aber andere Verfahren des maschinellen Lernens betrachtet, die eine automatisierte Reaktion auf Veränderungen ermöglichen sollen.

5.2 Verwendung von Knowledge Graph Embeddings

Um die beschriebenen Grenzen des regelbasierten Ansatzes zu umgehen, wird eine Methode benötigt, die automatisiert auf Veränderungen reagieren kann, ohne dass manuell Regeln festgelegt werden müssen. Gängige Verfahren des maschinellen Lernens benötigen dazu eine vektorielle Repräsentation des Graphen und der darin enthaltenen Knoten.

Es ist denkbar, händisch festzulegen, wie diese Repräsentation gestaltet sein soll. So könnte zum Beispiel für jeden Knoten die Anzahl der eingehenden und ausgehenden Kante pro Relationstyp gezählt werden. Ein Vorgehen, das nach händisch festgelegten Regeln eine Repräsentation erzeugt, wird als *Feature Engineering* bezeichnet. Eine durch Feature Engineering erzeugte Repräsentation führt allerdings zu einer Darstellung, deren Qualität in erster Linie von der Güte des Feature Engineering abhängt. Innerhalb der letzten sieben Jahre ist das automatisierte Lernen von Repräsentationen für Graphen (das sogenannte *Representation Learning* [Goo+16, S. 524 ff.]) in den Fokus der akademischen Community im Bereich des maschinellen Lernens gerückt. Beim Representation Learning für Graphen werden Abbildungen von Graphen in einen dichten Vektorraum gelernt. Ein Beispiel für einen im Zusammenhang mit Representation Learning verwendeten Vektorraum ist ein 300-Dimensionaler euklidischer Raum. Die Abbildung in den Vektorraum wird so gestaltet, dass sich mit der Abbildung des Graphen im

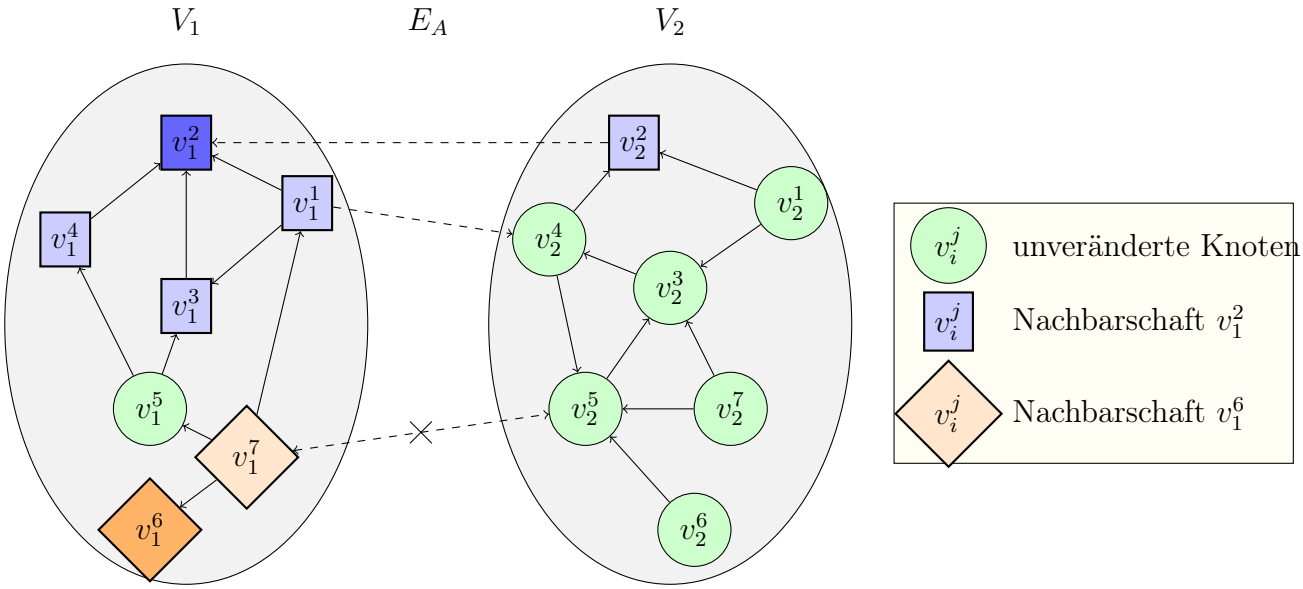

Abbildung 5.2: Nachbarschaft von Veränderungen

Raum verschiedene Aufgaben lösen lassen: Eine typische Aufgabe ist das Vorhersagen von weiteren Kanten oder Pfaden, die im Graph vorhanden sein sollten, aber bei der Erstellung des Graphen vergessen wurden. Diese Aufgabe wird als *Knowledge Base Completion*, *Knowledge Graph Completion* oder *Link Prediction* bezeichnet [Wan+14].

Die Grundidee des in diesem Abschnitt vorgestellten Ansatzes ist es, eine maschinell gelernte Repräsentation zu verwenden, um Vorschläge für Alignment-Anpassungen zu erzeugen. Dazu wird zunächst jedem Knoten des Graphen ein Punkt in einem Embedding-Raum, meistens dem $\mathbb{R}^n$ oder $\mathbb{C}^n$ zugewiesen. Anschließend werden mit Repräsentationen aus diesem Embedding-Raum mehrere Machine-Learning-Modelle auf Daten aus einer Versionsveränderung eines Graphen trainiert. Zu einer Versionsveränderung liegen sowohl der Graph vor der Änderung (G) und der Graph nach der Versionsveränderung (G') vollständig vor. Die trainierten Machine-Learning-Modelle können dann genutzt werden, um bei zukünftigen Veränderungen Vorschläge für die Anpassung des Alignments zu generieren. Dieses so entwickelte Verfahren wird dann auf einer weiteren Versionsveränderung evaluiert.

Um die Problemstellung dieser Arbeit mit maschinellem Lernen zu bearbeiten, muss die Formulierung des Problems angepasst werden. Die ursprüngliche Problemstellung aus Problem 4.3.1 kann mit Verfahren des maschinellen Lernens nicht vollständig direkt bearbeitet werden. Die Funktion, die dazu gelernt werden müsste, währe eine Funktion $(\mathcal{G}, \mathcal{P}(\mathcal{V})) \to \mathcal{P}(\mathcal{V})$, wobei $\mathcal{P}$ die Potenzmenge beschreibt. Sowohl der Problem- als auch Lösungsraum ist also sehr groß. Einerseits wären sowohl Eingabe als auch Ausgabe des Problems Mengen variabler Größe, was mit Methoden des maschinellen Lernens schwierig zu betrachten ist. Andererseits hätte

man nur sehr wenige Trainingsbeispiele, da es nur wenige dokumentierte Versionsveränderungen zwischen Ontologien gibt, die durch Alignments vernetzt sind. Ansätze wie zum Beispiel [Sut+14], die Sequenzen auf Sequenzen abbilden, wenden mehrere Strategien an, um mit diesen Herausforderungen umzugehen: Es werden sehr viele Beispiele für diese Mengen betrachtet, da ein geschickter Ansatz angewandt wird, um weitere Beispiele zu generieren, die beim Training verwendet werden können. Außerdem werden Informationen aus der Ordnung der Daten wiederverwendet. Im Kontext der Problemstellung dieser Arbeit gibt es nur wenige passende Beispiele, bei denen außerdem unklar ist, wie eine Ordnung gestaltet werden kann. Dementsprechend sind Methoden wie in [Sut+14] beschrieben nicht sinnvoll anwendbar. Grundsätzlich ist es also nur schwer möglich, alle Veränderungen einer Ontologie gleichzeitig zu betrachten. Es muss also eine Möglichkeit gefunden werden, die Problemstellung so zu formulieren, dass sie mit Methoden des maschinellen Lernens verarbeitet werden kann.

Um dennoch Methoden des maschinellen Lernens anwenden zu können, wird das Problem dieser Arbeit etwas kleinteiliger formuliert. Anstatt alle Veränderungen auf einmal zu betrachten, wird die Nachbarschaft von veränderten Knoten untersucht. Die in den Ontologien veränderten Knoten werden in zwei Kategorien aufgeteilt: (0) einerseits, die veränderten Knoten, in deren Nachbarschaft eine Alignment-Veränderung vorliegt, (1) andererseits die veränderten Knoten, bei denen keine Alignment-Veränderung in der Nachbarschaft vorliegt. Es wird also eine Funktion $V \to \{0, 1\}$ mit deutlich kleinerem Definitions- und Wertebereich gelernt. Durch diese kleinteiligere Betrachtung liegen außerdem viele kleine Beispiele anstatt wenige Große vor.

Ein Beispiel zur Verdeutlichung des Nachbarschaftskonzepts und der zugehörigen Kategorisierung ist in Abbildung 5.2 dargestellt. In diesem Beispiel werden zur einfachen Verständlichkeit jene Knoten zur Nachbarschaft eines Knotens gezählt, die einen Schritt von diesem Knoten entfernt sind. In dem dargestellten Beispiel wird die Nachbarschaft der Knoten v_1^2 und v_1^6 dargestellt, die in dem Beispiel veränderte Knoten repräsentieren sollen. Die konkrete Veränderung der Knoten ist nicht abgebildet. Als Konsequenz der Veränderung wurde eine Alignment-Kante zwischen v_1^7 und v_2^5 entfernt. Zur Nachbarschaft von v_1^2 gehören die Knoten v_1^1, v_1^3, v_1^4 und v_2^2. Keiner dieser Knoten liegt an einer veränderten Alignment-Kante. Zur Nachbarschaft von v_1^6 gehört nur v_1^7. v_1^7 liegt an einer veränderten Alignment-Kante. v_1^6 und v_1^2 würden also unterschiedlich kategorisiert werden, da in der Nachbarschaft von v_1^6 ein Knoten ist, der von einer Alignment-Veränderung betroffen ist. In der Nachbarschaft von v_1^2 ist das nicht der Fall. v_1^6 würde also mit (0) kategorisiert und v_1^2 mit (1). Auf alle anderen Knoten würde keine Kategorisierung angewandt, da sie nicht verändert wurden.

5.2.1 Konzeptionelle Datengrundlage

Als Datengrundlage für das Trainieren und Evaluieren des Machine-Learning-Modells dienen drei Graphen,

$$G(V, E, s, t, \Lambda), G'(V', E', s', t', \Lambda'), G''(V'', E'', s'', t'', \Lambda'')$$

und zugehörige Versionsabbildungsfunktionen

$$I_V : V \to V',\ I_E : E \to E', I'_V : V' \to V'',\ I'_E : E' \to E''.$$

Aus diesen Graphen werden eine Trainings- und eine Testmenge an veränderten Knoten extrahiert. Knoten aus V sind in der Trainingsmenge und Knoten aus V' in der Testmenge. Zusätzlich werden alle veränderten Knoten markiert: Für jeden veränderten Knoten wird bestimmt, ob sich durch die Änderung an dem Knoten ein Alignment-Statement verändert hat. Um dies zu beurteilen, wird untersucht, ob in der Nachbarschaft des Knotens ein Alignment-Statement verändert wurde. Dazu wird die folgende Markierungsfunktion definiert:

Definition 5.2.1 (Knotenklassifizierung) *Gegeben die Größe der Nachbarschaft $n \in \mathbb{N}$, zwei Graphen $G(V, E, s, t, \Lambda)$, $G'(V', E', s', t', \Lambda')$ mit Versionsabbildungsfunktionen $I_V : V \to V'$, $I_E : E \to E'$ und den von Alignment-Änderungen unter I betroffenen Knoten M. Dann ist die Klassifizierungsfunktion $\lambda_{G,G',I} : V \to \{\boxed{c}, \boxed{n}\}$ definiert als*

$$\lambda_{G,G',I}(v) = \begin{cases} \boxed{c} & \exists v_c \in M : v_c \in V_{v,n} \\ \boxed{n} & sonst. \end{cases}$$

Veränderte Knoten, die maximal n Kanten von einer Alignment-Veränderung entfernt sind, werden also mit $\boxed{c}$, alle anderen Knoten mit $\boxed{n}$ markiert. Die Markierung der Knoten bezieht sich dabei immer auf die Veränderung zur nächsten Graph-Version und damit die jeweilige Versionsabbildungsfunktion. Die Trainingsdaten werden durch veränderte Knoten aus V und die zugehörigen Markierungsfunktion $\lambda_{G,G',I}$ und die Testdaten werden durch veränderte Knoten aus V' und die zugehörigen Markierungsfunktion $\lambda_{G',G'',I'}$ beschrieben. Wurde also ein Knoten von G nach G' und ebenfalls von G' nach G'' verändert, kann dieser in der Test- und Trainingsmenge unterschiedliche Markierungen erhalten. Die durch das beschrieben Verfahren erstellten Daten können dann genutzt werden um ein Machine-Learning-Modell zu trainieren. Die Trainings- und Testdaten zu vertauschen könnte aus akademischer Sicht ebenfalls interessant sein, ist praktisch allerdings weniger relevant, da man so bereits in der Vergangenheit

liegende Veränderungen vorhersagen würde.

5.2.2 Embedding

Embeddings werden auf dem Graphen G' erzeugt, da dies der einzige Graph ist, der zum Zeitpunkt des Trainings des Machine-Learnings-Modells vollständig zur Verfügung steht. Das führt dazu, dass nicht alle Knoten aus G'' in den Embedding-Raum abgebildet werden können. Allerdings ist es ohnehin schwierig für neue Knoten Aussagen zu treffen, wenn man aus der bestehenden Graphstruktur lernen möchte. Graph-Embeddings können auf verschiedenste Weise erzeugt werden. Für den grundlegenden Ansatz ist allerdings nur relevant, dass sie ermöglichen, einem Knoten im Graph einen reellwertigen Vektor zuzuordnen, da sie so für andere Ansätze des maschinellen Lernens verwendet werden können. Dazu wird die folgende Spezialisierung von Definition 2.3.1 verwendet:

Definition 5.2.2 (Graph-Embedding) *Ein Graph-Embedding τ des Graphen $G(V, E, s, t, \Lambda)$ ist eine Abbildung von Graph-Knoten in den $\mathbb{R}^n$:*

$$\tau : V \to \mathbb{R}^n$$

n kann dabei frei gewählt werden.

Wie bereits in Abschnitt 2.3.1 aufgeführt werden in dieser Arbeit Embeddings für Wissensgraphen in drei Kategorien unterteilt: (1) Ansätze aus der Sprachmodellierung, (2) Knowledge-Graph-Completion-Ansätze mit Translativen Modellen und (3) Knowledge-Graph-Completion-Modelle mit Faktorisierung der Adjazenzmatrix. Eine Übersicht über die Scoring-Funkktion der Modelle angewandt auf den in dieser Arbeit verwendeten Formalismus zur Repräsentierung von Wissensgraphen ist in Tabelle 5.1 dargestellt.

(1) **Sprachmodellierung** In der ersten Kategorie steht ein Ansatz zur Verfügung: RDF2Vec [RP16]. RDF2Vec arbeitet auf zufälligen Pfaden durch den Graphen und betrachten diese Pfade dann als Sätze im Sinne eines Sprachmodells. Die in Tabelle 5.1 dargestellten Scoring-Funktionen beziehen sich darauf, die Wahrscheinlichkeit zu berechnen, dass eine Kante oder ein Knoten an einer bestimmten Stelle in einem Random Walk vorkommt. Die Scoring-Funktionen werden in Abschnitt 2.3.1 im Detail vorgestellt. Das Erzeugen von Random Walks führt dazu, dass insbesondere bei großen Graphen viel Rechenzeit und Speicherplatz für die Erstellung der Pfade benötigt wird. Dementsprechend können diese Verfahren nicht immer angewandt werden.

(2) **Translative Modelle** In der zweiten Kategorie gibt es sehr viele Ansätze, darunter TransE [Bor+13] und TransH [Wan+14]. Die in Tabelle 5.1 dargestellten Scoring-Funktionen

Tabelle 5.1: Graph-Embeddings: Überblick

	Modell	Scoring-Funktion
Sprachmodellierung		
	RDF2Vec [RP16]	$\frac{e^{(v^T v'_{w_t})}}{\Sigma_{w=1}^{V} e^{(\bar{v}^T v'_w)}}$
Translative Modelle		
	TransE [Bor+13]	$\|\|v_{s(e)} + w_{\Lambda(e)} - v_{t(e)}\|\|_n$
	TransH [Wan+14]	$\|\|v_{s(e,\perp,e} + d_{\Lambda e} - v_{t(e,\perp,e}\|\|_n$
Faktorisierung		
	Distmult [Yan+14]	$< w_{\Lambda(e)}, v_{s(e)}, v_{t(e)} >$
	Complex [Tro+16]	$Re(< w_{\Lambda(e)}, v_{s(e)}, v_{t(e)}) >$

werden genutzt um die Wahrscheinlichkeit des Auftretens einer Kante e zu bewerten. Es wurden die zwei populärsten Ansätze (nach Anzahl der Zitierungen) gewählt. Es wäre auch möglich, die Verfahren zu verwenden, die auf aktuellen Standard-Datensätzen zu Knowledge Base Completion nach gängigen Evaluationsmethoden am besten abschneiden. Allerdings bedeutet, dass diese Verfahren besser abschneiden, nicht unbedingt, dass die durch die Verfahren generierten Embeddings besser als Repräsentation für die Aufgabe dieser Arbeit geeignet sind. Typische Datensätze aus dem Bereich Knowledge Base Completion betrachten nur spezifische Domänen. Außerdem wird auf diesen Datensätzen evaluiert, wie gut sich neue Kanten im Graph vorhersagen lassen. Es ist also nicht unbedingt anzunehmen, dass Verfahren, die bei einer solchen Evaluation gut abschneiden auch gut geeignet sind, um Veränderungen im Zusammenhang mit Alignments darzustellen.

Außerdem ist es bei der Anzahl von vorliegenden Modellen nicht unwahrscheinlich, dass einige Modelle auf die Eigenheiten der Standard-Datensätze overfitten und für andere Arten von Datensätzen nicht geeignet sind. In [Kad+17] konnte gezeigt werden, dass durch Hyperparameter-Tuning von einfachen Modellen neuere Ansätze übertroffen werden konnten. Die neueren Ansätze waren den vorher veröffentlichten Ansätzen also nicht überlegen, sondern wurden nur besser an den Datensatz angepasst. Aus diesem Grund wurden in dieser Arbeit die populäreren und simpleren Modelle verwendet, nämlich TransE und TransH.

③ **Faktorisierung** Aus der dritten Kategorie werden DistMult [Yan+14] betrachtet, welches in [Kad+17] als Ansatz verwendet wurde, sowie ComplEx [Tro+16], ein Ansatz, der seit mehreren Jahren sehr gute Ergebnisse auf einem der zwei populären Datensätze liefert. Alle hier vorgestellten Ansätze außer ComplEx passen zu der in dieser Arbeit verwendeten Embedding-

Tabelle 5.2: Klassifikationsverfahren

Category	Method
Regression	Logistic Regression (LR)
Naive Bayes	Gaussian Naive Bayes (NB)
Nearest Neighbour	K Nearest Neighbour
Entscheidungsbäume	CART, Random Forest
Support Vector Machines	RBF-Kernel, Linear Kernel
Neuronale Netze	Multilayer Perceptron

Definition (Definition 5.2.2). ComplEx lernt eine Funktion $V \to \mathbb{C}^n$. Um diese Embeddings verwenden zu können, werden in dieser Arbeit die Real- und Imaginärteile der einzelnen Vektoren konkateniert, so dass ein Embedding $V \to \mathbb{R}^{2n}$ gelernt wird. So sind alle Ansätze in der Lage, einem Knoten im Graphen einen Punkt im $\mathbb{R}^n$ zuzuordnen.

5.2.3 Klassifikation

Durch die in Abschnitt 5.2.1 beschriebene Zuweisung von Labeln zu Knoten und die in Abschnitt 5.2.2 beschriebene Zuweisung von Knoten zu reelwertigen Vektoren können verschiedene Verfahren des maschinellen Lernens angewendet werden. Dadurch, dass jedem Knoten in den Trainingsdaten eine von zwei möglichen Markierungen zugeordnet wird, lassen sich Verfahren zur binären Klassifikation anwenden. Die Funktion, die dabei gelernt werden soll, ist eine Funktion $\gamma : \mathbb{R}^n \to \{\boxed{c}, \boxed{n}\}$.

Typischerweise werden Verfahren zur binären Klassifikation in die Kategorien *Regression, Naive Bayes, Nearest Neighbour-Verfahren, Entscheidungsbäume, Support Vector Machines* und *Neuronale Netze* eingeteilt [Goo+16, S. 137 ff.]. Die in dieser Arbeit verwendeten Verfahren sind in Tabelle 5.2 aufgelistet. Diese Verfahren wurden so ausgewählt, dass aus jeder Kategorie der Klassifikationsverfahren mindestens ein Verfahren verwendet wird und die Verfahren möglichst in populären Machine-Learning-Libraries implementiert sind. Die einzelnen Verfahren sind jeweils in Abschnitt 2.3.2 näher beschrieben.

Das Verfahren zum Trainieren des Klassifikators ist in Algorithmus 7 dargestellt. Im ersten Schritt wird die Embedding-Abbildung τ trainiert. Danach werden die Trainingsdaten T erstellt, indem jedem veränderten Knoten $c \in C$ ein Vektor durch τ und ein Label durch λ zugewiesen werden. Mit diesen Trainingsdaten wird die Klassifikationsfunktion γ trainiert, welche dann angewendet werden kann.

Um γ anzuwenden, muss für einen Knoten, der klassifiziert werden soll, die Embedding-Funktion τ ausgewertet werden. Das Ergebnis dieser Funktion ist Eingabe für γ. Das führt dazu, dass Knoten, die nicht in V' enthalten sind, nicht klassifiziert werden können. Dieser Nachteil wird in Kauf genommen, da für Knoten, die nicht in V' enthalten sind, kein Embedding in einer konsistenten Graph-Version bestimmt werden kann.

Algorithmus 7 Klassifikation mit Graph-Embeddings

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Graph $G'(V', E', s', t', \Lambda')$
Input: Funktion $I_V : V \to V'$
Input: Funktion $I_E : E \to E'$
Input: Funktion $\lambda : V \to \{\boxed{c}, \boxed{n}\}$
procedure ERSTELLEN DES KLASSIFKATORS
  trainiere $\tau : V \to \mathbb{R}^n$ auf Trainingsdaten G'
  $C \leftarrow$ Veränderte Knoten C von G nach G' $(C \subseteq V)$
  $T \leftarrow \emptyset$
  for all c in C **do**
    $T \leftarrow T \cup (\tau(c), \lambda_{G,G',I}(c))$
  end for
  trainiere $\gamma : \mathbb{R}^n \to \{\boxed{c}, \boxed{n}\}$ auf Trainingsdaten T
  return γ
end procedure

5.2.4 Anwendbarkeit des Ansatzes

Wie auch für den auf Inferenzen und Regeln aufbauenden Ansatz kann auch für diesen Ansatz untersucht werden, welche Voraussetzungen für die Anwendbarkeit dieses Ansatzes gelten. Die zentrale Einschränkung, die für diesen Ansatz gegeben ist, ist das wie auch im vorherigen Fall, Eingabedaten vorhanden sein müssen (siehe Vorbedingung 5.1.1).

Zusätzlich müssen Trainingsdaten vorhanden sein, die so annotiert sind, dass ein Klassifikationsalgorithmus diese Trainingsdaten verwenden kann.

Vorbedingung 5.2.1 (Trainingsdaten) *Die Funktion $\lambda_{G,G',I} : V \to \{\boxed{c}, \boxed{n}\}$ ist gegben.*

Des weiteren muss ein Verfahren gegeben sein, mit dem Embeddings für die Graphen G eine Embedding-Funktion erzeugt werden kann. Dem entsprechend müssen die Daten klein genug sein, dass ein Embedding erzeugt werden kann.

Vorbedingung 5.2.2 (Embeddings) *Ein Verfahren zum Lernen von $\tau : V' \to \mathbb{R}^n$ ist gegeben.*

Außerdem muss eine Implementierung eines Klassifikationsverfahrens gegeben sein.

Vorbedingung 5.2.3 (Klassifikationsverfahren) *Ein Verfahren zum Lernen von* $\gamma : \mathbb{R}^n \rightarrow \{\boxed{c}, \boxed{n}\}$

In der Praxis bedeutet das, dass zwei Versionen von den Ontologien und dem Alignment gegeben sein müssen, um das Verfahren trainieren zu können. Daraus lässt sich (wenn die Daten klein genug sind beziehungsweise die entsprechenden Ressourcen zur Verfügung stehen) ein Embedding erzeugen. Außerdem lässt sich aus den zwei Ontologieversionen G und G' die Funktion $\lambda_{G,G'}$ berechnen.

5.3 Kombination von maschinellem Lernen mit Veränderungsmodellen

Die bisher vorgestellten Ansätze betrachten entweder ausschließlich ein durch Regeln formalisiertes Modell von Veränderungen oder eine gelernte Repräsentation der Graphstruktur. Diese Konzepte modellieren Informationen, die jeweils wenige Überschneidungen enthalten. Es erscheint also vielversprechend, sie zu verbinden. So können möglichst viele Informationen betrachtet werden, die zur Anpassung von Alignments an Ontologieveränderungen nützlich sind. Ein Ansatz für ein Vorgehen, dass sowohl ein Veränderungsmodell als auch eine gelernte Repräsentation der Graphstruktur mit einbezieht, wird in diesem Abschnitt vorgestellt.

Die Grundgedanken eines Ansatzes, der ein Veränderungsmodell in einen Ansatz mit maschinellem Lernen einbezieht sind in Abbildung 5.3 dargestellt. In der Abbildung sind Aspekte, die bereits bei der Verwendung von Embeddings betrachtet wurden in grün dargestellt, die neuen Aspekte sind in lila abgebildet. Für den Ansatz wird die in Abschnitt 5.2.1 vorgestellte konzeptionelle Datengrundlage verwender. Die Kernidee des Ansatzes ist es, eine gemeinsame Repräsentation für die Graphstruktur und die Art der Veränderung für jeden veränderten Knoten zu finden. Dazu wird ein Werkzeug zur Erkennung von Veränderungen im Rahmen eines Veränderungsmodells verwendet. Eingabe für dieses Werkzeug sind die Ontologien in unveränderter (V_1, V_2) und veränderter Form (V_1', V_2'). Das Ergebnis wird für jeden Knoten in eine Vektorrepräsentation (*Change Rerpresentation*) überführt. Diese Repräsentation stellt für jeden veränderten Knoten dar, welche Art von Veränderung am Knoten vorgenommen wurde. Wie bereits bei den Verfahren, die Embeddings verwenden, wird über dem gesamten Graph G ein gemeinsames Embedding berechnet. So liegen für jeden Knoten zwei Vektoren vor: Einer, der die Art der Veränderungen und einer, der die Graphstruktur repräsentiert. Diese Vektoren

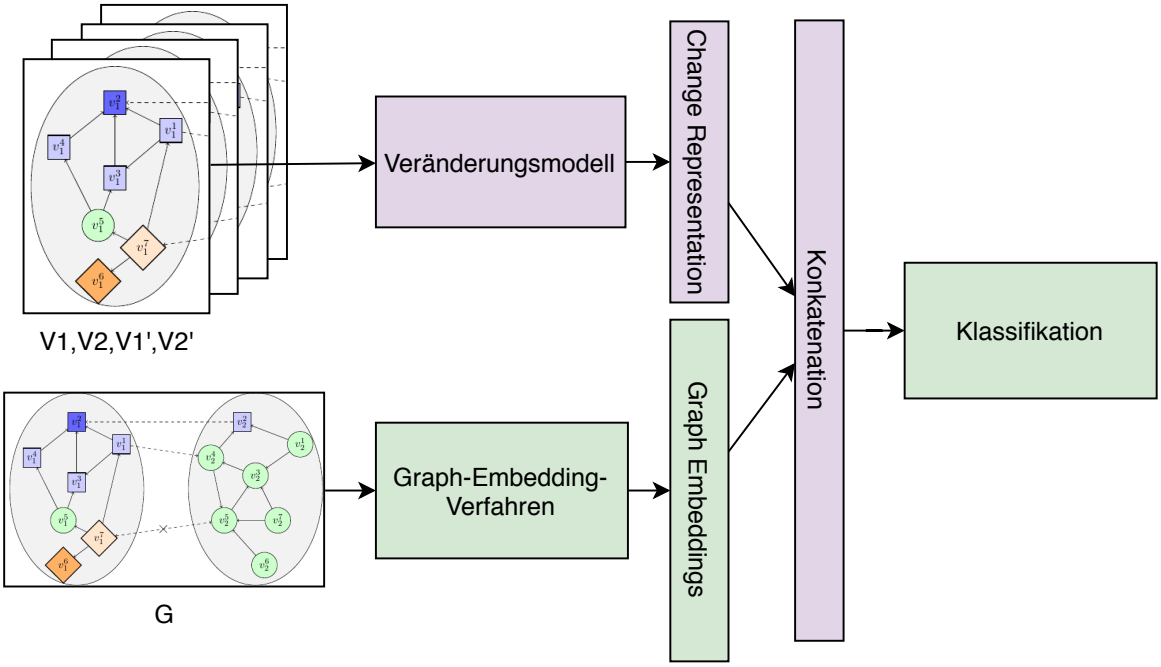

Abbildung 5.3: Ansatz zur Verwendung eines Veränderungsmodells mit Embeddings

werden durch eine Konkatenation zusammengeführt und in der zusammengeführten Form als Eingabe für die Klassifikation verwendet.

5.3.1 Veränderungsmodelle

Veränderungsmodelle werden in dieser Arbeit als Klassifizierungssysteme betrachtet. Aufgabe eines Veränderungsmodells ist es, zu bestimmen, welcher Art von Veränderung ein Knoten durchlaufen hat. Ein Veränderungsmodell wird formal so gefasst, dass zu jeder Veränderungsklasse eine natürliche Zahl existiert, die diese Klasse identifiziert. Für die Formulierung des Ansatzes selbst ist es unerheblich, wie die Veränderungsmodelle genau funktionieren. Dementsprechend werden die Veränderungsmodelle hier als gegeben betrachtet.

Definition 5.3.1 (Veränderungsmodell) *Ein Veränderungsmodell Ψ ist eine Menge von Klassen von möglichen Veränderungen Ψ in einem Graph. Jeder Klasse $m \in \Psi$ wird ein eindeutiger Index $i \in \mathbb{N}$ mit $0 \leq i < |\Psi|$ zugeordnet.*

Um Veränderungen Klassen zuzuweisen, wird eine Funktion für jede Veränderungsklasse definiert. Diese Funktion ist keine Labeling-Funktion, sondern ist eher an eine charakterisitische Funktion angelehnt. Es wird also nicht einem Knoten ein Veränderungslabel zugeordnet, sondern für jede Klasse existiert eine Funktion, die je nachdem ob ein Knoten zu einer Veränderungsklasse gehört dem Knoten den Wert 0 oder 1 zuordnet.

Definition 5.3.2 (Veränderungstypfunktion) *Die Veränderungstypfunktion* $\delta_i : V \to (0, 1)$ *bildet einen veränderten Knoten* $c \in V$ *aus dem Graphen* $G(V, E, s, t, \Lambda)$ *wie folgt ab:*

$$\delta_i(c) \begin{cases} 1 & \textit{wenn } c \textit{ sich von } G \textit{ nach } G' \textit{ so verändert hat, dass die} \\ & \textit{durch } i \textit{ bezeichnete Veränderung vorliegt} \\ 0 & \textit{sonst.} \end{cases}$$

Mit der Veränderungstypfunktion lässt sich dann ein Vektor erzeugen, der die Veränderungen an einem Knoten repräsentiert. Dieser Veränderungsvektor dient als One-Hot-Encoding der Veränderungen.

Definition 5.3.3 (Veränderungstypvektorfunktion) *Die Veränderungstypvektorfunktion* $\delta : V \to \{0, 1\}^{|\Psi|}$ *bei gegebenem Veränderungsmodell* Ψ *ist definiert als*

$$\delta(v) = \begin{pmatrix} \delta_0(v) \\ \delta_1(v) \\ .. \\ \delta_{|\Psi|}(v) \end{pmatrix}$$

5.3.2 Klassifikation

Wie bereits in Abschnitt 5.2.3 beschrieben, wird bei der Klassifikation eine Funktion $\gamma : \mathbb{R}^n \to \{\boxed{c}, \boxed{n}\}$ gelernt. Dementsprechend wird als Eingabe ein Vektor als Repräsentation eines veränderten Knotens benötigt. Für einen veränderten Knoten v liegen allerdings zwei Vektoren, $\delta(v)$ und $\tau(v)$, vor. Eine Möglichkeit, daraus eine Eingabe für einen Klassifikationsalgorithmus zu erzeugen, ist, diese Vektoren zu konkatenieren. Dies ist auch der Ansatz, der in dieser Arbeit verfolgt wird.

Auf diese konkatenierten Vektoren kann dann ein Klassifikationsverfahren angewandt werden. Der vollständige Algorithmus zum Erstellen eines Klassifikators ist in Algorithmus 8 dargestellt. Für die Vektorkonkatenation wird in dieser Arbeit das Symbol $\oplus$ verwendet. Der Algorithmus unterscheidet sich an nur wenigen Stellen von Algorithmus 7: Als Eingabe wird zusätzlich ein Veränderungsmodell benötigt, das für die neue Funktion zur Erzeugen von Veränderungsrepräsentationen υ ausgewertet werden muss. Aus dieser Auswertung wird die Funktion υ erzeugt. Außerdem müssen für das Erstellen der eigentlichen Trainingsdaten die Konkatenation des Embeddings und der Veränderungsrepräsentation berechnet werden, ansonsten funktionieren die Algorithmen 7 und 8 gleich.

Bei der Anwendung der Klassifkationsfunktion muss nun ebenfalls etwas anders vorgegangen werden, als wenn nur Embeddings betrachtet werden: zusätzlich dazu, dass das Embedding

eines veränderten Knoten mit der Funktion τ berechnet werden muss, muss eine Veränderungsrepräsentation berechnet werden, wenn eine neue Veränderung klassifiziert werden soll. Dafür muss das gleiche Veränderungsmodell verwendet werden, dass beim Erzeugen der Trainingsdaten genutzt wurde.

Algorithmus 8 Klassifikation mit Graph-Embeddings und Veränderungsmodell

Input: Graph $G(V, E, s, t, \Lambda)$
Input: Graph $G'(V', E', s', t', \Lambda')$
Input: Funktion $I_V : V \to V'$
Input: Funktion $I_E : E \to E'$
Input: Veränderungsmodell Ψ
Input: Funktion $\lambda : V \to \{\boxed{c}, \boxed{n}\}$
procedure ERSTELLEN DES KLASSIFKATORS
  trainiere $\tau : V \to \mathbb{R}^n$ auf Trainingsdaten G'
  erstelle $\upsilon : V \to \{0,1\}^{|\Psi|}$ aus Veränderungen von G nach G'
  $C \leftarrow$ Veränderte Knoten von G nach G' $(C \subseteq V)$
  $T \leftarrow \emptyset$
  for all c in C **do**
    $T \leftarrow T \cup (\tau(c) \oplus \upsilon(c), \lambda_{G,G',I}(c))$
  end for
  trainiere $\gamma : \mathbb{R}^{n+|\Psi|} \to \{\boxed{c}, \boxed{n}\}$ auf Trainingsdaten T
  return γ
end procedure

5.3.3 Anwendbarkeit des Ansatzes

Für diesen Ansatz müssen alle in Abschnitt 5.2.4 aufgeführten Vorbedingungen erfüllt sein. Das sind die Vorbedingungen 5.1.1, 5.2.1, 5.2.2 und 5.2.3. Zusätzlich muss noch ein Verfahren vorhanden sein, dass Veränderungen zwischen den Ontologien Klassen aus einem Veränderungsmodell zuordnet.

Vorbedingung 5.3.1 *Das Veränderungsmodell Ψ und die Funktion $\delta_i : V \to \{0,1\}^{|\Psi|}$ sind gegeben.*

Für eine Anwendung in der Praxis kann die Funktion δ_i durch eine Software zur Erkennung von Veränderungen in Ontologien berechnet werden. Dazu müssen zwei Ontologieversionen vorhanden sein.

5.4 Verwendung von Graph Convolutional Networks

Der in den vorherigen Abschnitt beschriebene Embedding-basierte Ansatz hat mindestens einen relevanten Nachteil: die Repräsentation der Knoten im Graph wird so optimiert, dass die Aufgabe, für den diese Verfahren eigentlich vorgesehen sind, gut durchgeführt werden kann – diese Repräsentation ist dann dementsprechend auch an diese Aufgabe angepasst. Die gelernte Repräsentation der Knoten ist also darauf optimiert, weitere Kanten im Graph vorherzusagen und nicht das in dieser Arbeit behandelte Problem zu lösen.

Es ist denkbar, dass bessere Ergebnisse erzielt werden können, wenn die Repräsentation direkt an die zu bearbeitende Aufgabe angepasst ist. Es ist also im Kontext dieser Arbeit wünschenswert, wenn die Repräsentation der Knoten so gestaltet wird, dass sich damit möglichst gut vorhersagen lässt, welche Anpassungen am Alignment vorgenommen werden müssen. Nicht relevant jedoch ist die Vorhersage von zusätzlichen Kanten im Graph, wie das bei den meisten Verfahren zum Embedding von Graphen üblich ist.

Ein weiterer wichtiger Aspekt ist, dass bei der Verwendung von Embeddings Informationen verloren gehen. Letztendlich bildet ein Embedding nur eine Approximation der Nachbarschaft eines Knoten ab. Es ist also denkbar, dass genau die Information, die zur Vorhersage von veränderungsbedürftigen Alignment-Statements gebraucht würde, nicht erfasst werden. Außerdem werden beim Embedding von Graphen Ressourcen verschwendet: Typische Graph-Embedding-Methoden berechnen ein Embedding für jeden Knoten im Graph. Da der Großteil der Knoten aber für die in dieser Arbeit betrachteten Aufgabenstellung nicht relevant ist, ist ein Großteil der Embeddings nicht weiter nützlich.

Eine Alternative zur getrennten Berechnung der Embeddings stellen neuronale Netze dar, die direkt auf der Graphstruktur arbeiten. Diese Netze integrieren die Struktur des Graphen, auf dem sie arbeiten direkt in die Topologie des neuronalen Netzes. Die Struktur des Graphen ist also vollständig abgebildet. Im Gegensatz zu Embedding-basierten Ansätzen wird also nicht nur eine Approximation betrachtet, sondern der tatsächliche Graph. Dementsprechend ist die Funktion die zur Klassifikation gelernt wird, anders als in Abschnitt 5.2.2, eine Funktion $\gamma : (V) \to \{\boxed{c}, \boxed{n}\}$. Es wird also anstatt zuerst eine Funktion $\tau : V \to \mathbb{R}^n$ und dann darauf aufbauend eine Funktion $\gamma : \mathbb{R}^n \to \{\boxed{c}, \boxed{n}\}$ zu lernen, nur eine Funktion gelernt, die vorher durch eine Komposition $\kappa = \gamma \circ \tau$ abgebildet wurde.

5.4.1 Modell

Aufgrund der genannten Vorteile wird dieser Ansatz parallel zu den anderen vorgestellen Ansätzen in dieser Arbeit verfolgt. Es wird ein *Relational Graph Convolutional Network* verwendet,

um den bereits beschriebenen Klassifikationstask umzusetzen. Ein Relational Graph Convolutional Network [Sch+18] ist eine Erweiterung von Graph Convolutional Networks [KW17]. Ein Graph Convolutional Network ist ein neuronales Netzwerk, in dem in jedem Layer des Netzes ein Neuron für jeden Knoten im Graph existiert. Die Graph-Struktur bestimmt dabei, wie die Neuronen zwischen den Layern verbunden sind: Sind zwei Knoten v_1 und v_2 durch eine Kante im Graph verbunden, existiert von einem Layer zum nächsten eine Verbindung der Neuronen, die jeweils v_1 und v_2 repräsentieren. Das führt zu folgender Regel für die Übertragung der Aktivität von einem Layer ins nächste:

$$h_i^{(l+1)} = \sigma\Big(\sum_{j \in V_{i,1}} \frac{1}{n} W^{(l)} h_j^{(l)}\Big) \tag{5.1}$$

Dabei ist $h_i^{(l)} \in \mathbb{R}^f$ die Aktivität im i-ten Knoten des l-ten Layers ($l \in \mathbb{N}$) des Netzes, $f \in \mathbb{N}$ die Dimensionalität der Knoten-Repräsentation, $V_{i,1}$ die Menge der Knoten, die direkt mit dem i-ten Knoten verbunden sind, $W^{(l)} \in \mathbb{R}^{f \times f}$ die Gewichtsmatrix im l-ten Layer, $n \in \mathbb{R}_+$ eine Normalisierungskonstante und $\sigma : \mathbb{R}^f \to \mathbb{R}^f$ eine Aktivierungsfunktion.

Die Regel zur Übertragung von Aktivität zwischen Layern aus Graph Convolutional Networks lässt sich auf Graphen mit gelabelten Kanten übertragen. Dazu wird für jedes Element des Wertebereichs R der Labeling-Funktion Λ eine eigener Graph gebildet und dafür Gleichung 5.1 berechnet und die Ergebnisse addiert. Zu diesem Wert wird noch die aktuelle Aktivität des Knoten, $h_i^{(l)}$, multipliziert mit einer Gewichtsmatrix $W_0^{(l)}$, addiert. Dann wird die Aktivierungsfunktion σ angewandt [Sch+18]:

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{j \in V_{i,1}^r} \Big(\frac{1}{n_{i,r}} W_r^{(l)} h_j^{(l)}\Big) + W_0^{(l)} h_i^{(l)}\right) \tag{5.2}$$

Dabei ist $W_r^{(l)}$ die für das Kantenlabel r spezifische Gewichtsmatrix in Layer l. $V_{i,1}^r$ ist die Menge aller mit i über Kanten mit dem Label r verbundenen Knoten und $W_0^{(l)}$ die Gewichtsmatrix für die Übertragung der Aktivität aus dem Knoten in sich selbst. Diese Layer können beliebig gestapelt werden und bilden so eine Funktion $\mathbb{R}^{l \times |V|} \to \{\boxed{c}, \boxed{n}\}$, wenn als letztes Layer ein Softmax-Layer verwendet wird. Wird als Eingabe nur der Feature-Vektor eines Knoten verwendet und die Eingabematrix ansonsten leer gelassen, lässt sich die Funktion $\kappa : V \to \{\boxed{c}, \boxed{n}\}$ abbilden.

5.4.2 Training

Mit dem im Bereich der Klassifikation üblichen Cross-Entropy Loss können die RGCN-Layer verwendet werden. Um als Klassifikator verwendet zu werden, kann ein RGCN-Klassifikator so verwendet werden wie in Algorithmus 7 beschrieben. Dabei werden allerdings keine Embeddings benötigt. Der Cross-Entropy-Loss angewandt auf diese Daten ist

$$\mathcal{L} = -\sum_{i=1}^{|T|}\sum_{v \in V} t_{iv} ln(h_{iv}^{(L)}) \tag{5.3}$$

Dabei ist T die Menge der Trainingsdaten, die entsprechend Algorithmus 7 erstellt wurden. V ist die Menge der Knoten im Graph. t_{iv} ist das Label, das dem Knoten v bei i-ten Trainingsbeispiel zugewiesen wurde und h_{iv}^{L} die Aktivität im letzten Layer im Knoten v für das i-te Trainingsdatum. Durch die Optimierung dieser Loss-Funktion erhält man einen Klassifikator, der ähnlich wie die im vorherigen Abschnitt beschriebenen Klassifikatoren arbeitet. Dabei verwendet dieser Klassifikator allerdings keine Embeddings, sondern bildet die Graphstruktur direkt ab.

5.4.3 Anwendbarkeit des Ansatzes

Die Anwendbarkeit von Relational Graph Convolutional Networks hat ähnliche Vorbedingungen wie die Anwendung der Ansätze, die Embeddings einsetzen. Allerdings können einige Einschränkungen fallengelassen werden. Erhalten bleiben die Vorbedingungen, dass Eingabedaten vorhanden sein müssen (Vorbedingung 5.1.1) und, dass gelabelte Trainingsdaten vorhanden sein müssen (Vorbedingung 5.2.1). Alle anderen Einschränkungen müssen für Relational Graph Convolutional Networks nicht erfüllt sein.

5.5 Verwendung der Ansätze

In diesem Kapitel wurden vier Ansätze zur Anpassung von Alignments an Ontologie-Veränderungen vorgestellt:

1. Ein regelbasierter Ansatz, der Inferenzen mit einbezieht (siehe Abschnitt 5.1).

2. Ein Ansatz, der zuerst eine Repräsentation des Graphen lernt und dann mit gängigen Verfahren des maschinellen Lernens vorhersagt, welche Veränderungen Änderungen am Alignment nach sich ziehen (siehe Abschnitt 5.2).

3. Ein Ansatz, der die Graph-Repräsentation mit einem Veränderungsmodell verbindet (siehe Abschnitt 5.3).

4. Ein Ansatz, der unter Verwendung eines Graph-Netzwerks die Repräsentation und die Klassifikation in einem Schritt lernt (siehe Abschnitt 5.4).

Im folgenden wird diskutiert, wie diese Ansätze implementiert werden können und eine Evaluation vorgestellt. Die Implementierung aller Ansätze soll durch ein gemeinsames Framework unterstützt werden. Bei der Evaluation müssen die Ansätze allerdings getrennt betrachtet werden, da die Voraussetzungen zur Anwendung sich unterscheiden. Die Evaluation des ersten Ansatzes wird anhand eines Beispiels aus dem Bereich der Interweaving Systems evaluiert, der zweite, dritte und vierte anhand eines Beispiels aus biomedizinischen Ontologien.

Kapitel 6

Entwurf und Umsetzung

In diesem Kapitel werden einige Aspekte der zur Untersuchung der in Kapitel 5 vorgestellten Ansätze erstellten Implementierung präsentiert. (Da eine vollständige Betrachtung der Implementierung an dieser Stelle den Rahmen sprengen und zur Betrachtung des Ansatzes und der Evaluation nicht viel beitragen würde werden hier nur Ausschnitte dargestellt). Es werden die zentralen Komponenten vorgestellt. Zunächst wird dazu beschrieben, wie genau Veränderungen in Ontologien erkannt werden. Als Umsetzung des in Abschnitt 5.1 vorgestellten Ansatzes wurde ein Framework entwickelt, dass sowohl die Berechnung von Inferenzen als auch die Reaktion auf Veränderungen kapselt. Diese Implementierung wird in Abschnitt 6.1 vorgestellt. Auch zur Betrachtung der in den Abschnitten 5.2, 5.3 und 5.4 vorgestellten Ansätze, die maschinelles Lernen einsetzen, wurden Implementierungsarbeiten durchgeführt. Die Architektur dieser Implementierung wird in Abschnitt 6.2 präsentiert.

6.1 Regelbasierter Umgang mit Veränderungen

An dieser Stelle wird die Implementierung des in Abschnitt 5.1 beschriebenen Ansatzes vorgestellt. Dazu greifen verschiedene Software-Komponenten ineinander. Der Zusammenhang der Komponenten ist in dem in Abbildung 6.1 vorgestellten FMC-Diagramm [Pyh07] dargestellt. Als Eingabe stehen zwei Versionen des vollständigen Graphen G und $\tilde{G}$ und der Parameter n zur Verfügung. Zuerst müssen die Veränderungen in den Ontologien erkannt werden, dies wird mithilfe einer Erweiterung des Plugins owl-diff umgesetzt.

Die Implementierung des eigentlichen Ansatzes verwendet die Ausgabe dieser Plugin-Erweiterung und wurde in der Form eines Scala-Frameworks umgesetzt (Diff-Inferenz-Framework). Dieses Framework zur automatisierten Anpassung von Alignments besteht aus zwei Teilen: Der Diff-Reasoner-Komponente, die zur Berechnung von Δ_{add} und Δ_{del} verwendet wird, und der

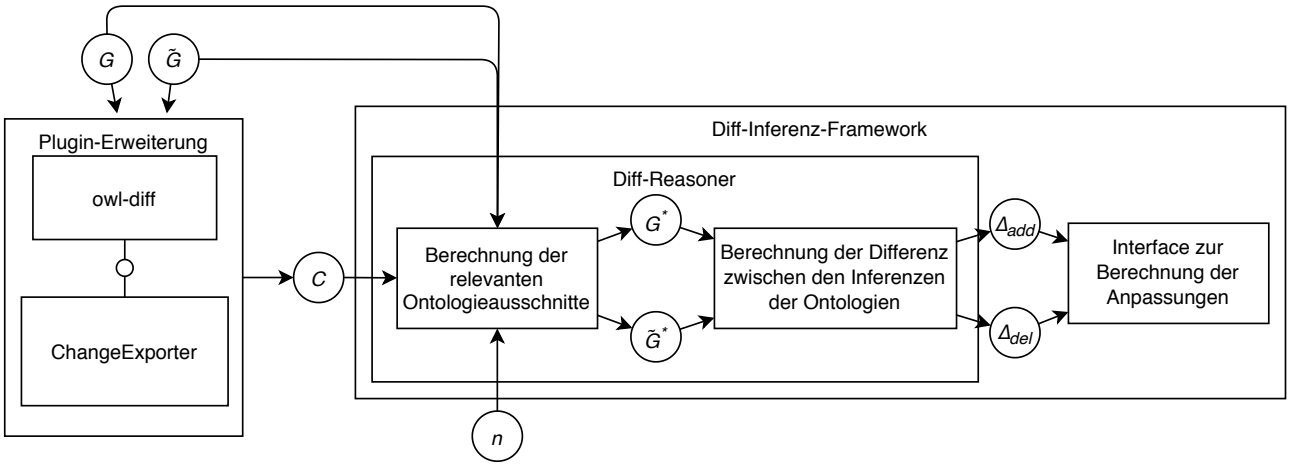

Abbildung 6.1: Zusammenhang der Komponenten zur Implementierung des regelbasierten Ansatzes

Komponente, die diese Berechnung mit dem Interface für die Funktionen α und γ verbindet.

Das Framework wurde vollständig mit Scala [Ode+08] umgesetzt. Scala wurde ausgwählt, da Scala Paradigmen des funktionalen Programmierens unterstützt und den Zugriff auf Java-Libraries ermölicht. Die wichtigsten frei verfügbaren Libraries zur Verarbeitung von Ontologien sind alle in Java geschrieben, dementsprechend ist dies eine wichtige Eigenschaft. Auf das funktionale Programmierparadigma wurde wert gelegt, da viele Reasoning-Frameworks mit Iteratoren arbeiten, die mit funktionalen Methoden sehr kompakt verarbeitet werden können. Diese Komponenten des Frameworks und die Erweiterung des Plugins werden im folgenden beschrieben.

6.1.1 Erweiterung des owl-diff-Plugins

Um die Veränderungen in Ontologien betrachten zu können und mit Datensätzen umgehen zu können, wird ein Werkzeug benötigt, das in der Lage ist, Veränderungen in Ontologien zu erkennen, zu kategorisieren und so abzulegen, dass diese Ergebnisse weiter verarbeitet werden können. Zum Zeitpunkt der Erstellung dieser Arbeit war das einzige frei verfügbare Werkzeug zur Erkennung von Veränderungen in Ontologien das Protégé-Plugin owl-diff [RN11]. Dieses Plugin ermöglicht allerdings keinen Export der Veränderungen in maschinenlesbarer Form. Aus diesem Grund wurde das Plugin um diese Funktionalität erweitert.

owl-diff produziert im wesentlichen eine Übersicht über Veränderungen zwischen zwei Ontologieversionen. Diese Veränderungen sind nach dem Konzept, das angepasst wurde, geordnet. Diese Ansicht wurde so erweitert, dass die angezeigte Übersicht exportiert werden kann, in dem ein Button hinzugefügt wurde, der eine Export-Komponente anstößt. Die um diesen Button zum Export der Veränderungen erweiterte Ansicht, die den Kern von owl-diff darstellt, ist in

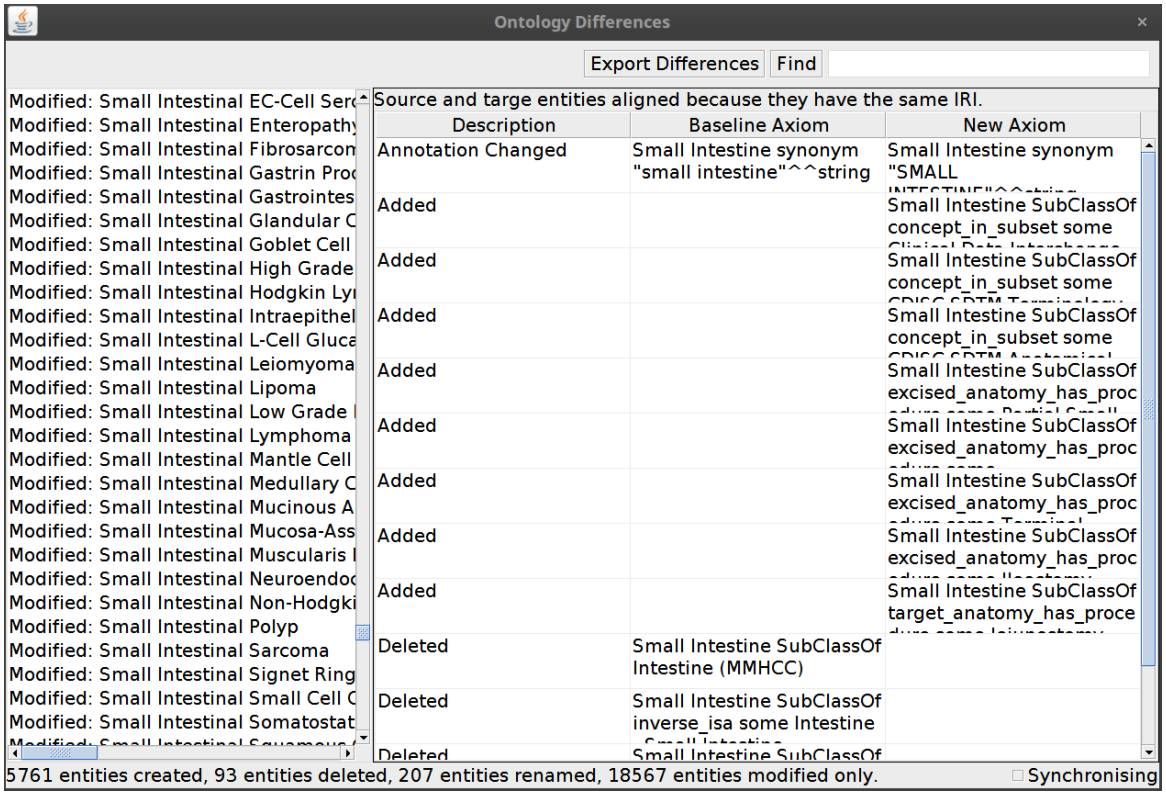

Abbildung 6.2: Screenshot des erweiterten OWL-Diff-Plugins

Abbildung 6.2 dargestellt. Auf der linken Seite sind die Namen der veränderten Konzepte zu sehen, auf der rechten Seite sind die Veränerungen an einem ausgewählten Konzept abgebildet. Oben in der Abbildung ist neben dem „Find“-Button der neue „Export Differences“-Button dargestellt.

Technisch bietet die dargestellte Benutzer-Oberfläche eine Sicht auf eine Modell-Klasse an, die Ergebnisse des dahinterliegenden Ontologie-Vergleichs kapselt. Die Klasse repräsentiert immer die Veränderung eines Konzepts in der Ontologie. Zur Repräsentierung dieser Veränderung werden die IRI des Konzepts in der Vorherigen und der neuen Version gespeichert. Des weiteren wird eine Collection von Axiom-Veränderungen gespeichert. Die Axiom-Veränderungen sind *Added* (Hinzufügen von Axiomen), *Deleted* (Entfernen von Axiomen), *Annotation Changed* (Veränderungen von Annotationen einer Klasse), und *Superclass Changed* (Verändern der Oberklasse).

Da die nötige Erweiterung sehr klein ist, wird sie direkt in der Benutzeroberfläche umgesetzt. owl-diff ist als java-plugin implementiert, dabei wird die Benutzeroberfläche mit dem swing-framework umgesetzt. Die Architektur der Umsetzung der Erweiterung ist in Abbildung 6.3 dargestellt. Bestehende Komponenten sind in orange, die Erweiterung in blau dargestellt. Die zentrale Komponente der UI von owl-diff ist der `Finder`. Diese Komponente greift auf die

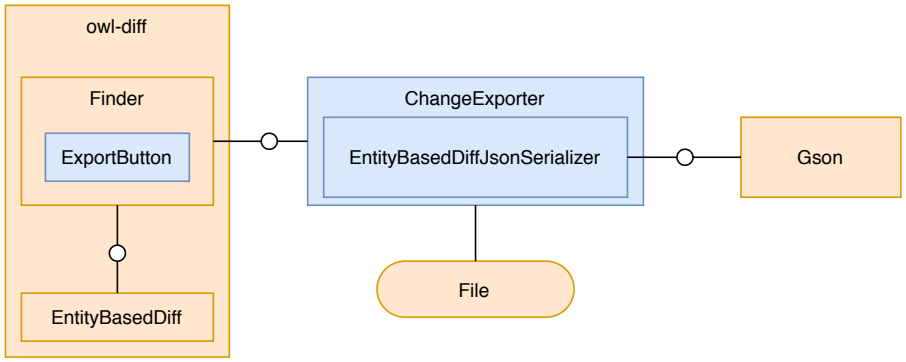

Abbildung 6.3: Architektur der Erweiterung des owl-diff-Plugins

Komponente `EntityBasedDiff` zu, um Veränderungsinformationen anzuzeigen. In den `Finder` wurde ein neuer Button, der `ExportButton`, eingefügt, der bei einem Klick den `ChangeExporter` aufruft. Diese Komponente verwendet das Framework Gson [Goo08], um die Informationen, die in `EntityBasedDiff` gespeichert sind, als JSON zu serialisieren. Dazu wird die Klasse `EntityBasedDiffJsonSerializer` verwendet. Diese nutzt das `File`-Interface, um die Daten in eine Datei zu schreiben.

6.1.2 Diff-Reasoner-Komponente

Die zentrale Aufgabe der Diff-Reasoner-Komponente ist es, zunächst die partielle Inferenz passend zur Definition 5.1.4 von zwei Ontologie-Versionen zu berechnen und dann daraus die Hinzugekommenen und entfernten Inferenzen Δ_{add} und Δ_{del} nach Definition 5.1.5 zu berechnen. Damit werden die ersten vier Zeilen des in Algorithmus 1 beschriebenen Verfahren umgesetzt. Dazu wird die Berechnung der partiellen Inferenz in zwei Aspekte aufgeteilt: (1) das Berechnen der relevanten Ontologieausschnitte (also die Nachbarschaft der Veränderten Knoten nach Definition 5.1.3) und (2) das Berechnen der Differenzen auf Basis der ausgewählten Ausschnitte der Ontologien.

Das Zusammenspiel dieser Aufgaben ist in Abbildung 6.4 dargestellt. In der Implementierung ist dieser Prozess als Methode mit dem Namen `inferenceWithReducedOntologies` umgesetzt. Diese Methode erhält als Eingabe die Graphen G und $\tilde{G}$, die Zahl n, die beschreibt, wie viele Schritte um einen veränderten Knoten die Ontologie noch als relevant betrachtet werden und eine Menge von veränderten Knoten C, die mit dem im vorherigen Abschnitt beschriebenen Plugin owl-diff erstellt wurde. Als API zur Repräsentierung der Graphen wird das Jena-Framework [Apa20] verwendet. Der Aspekt zur Berechnung der relevanten Ontologie wird in der mit (1) markierten Komponente durchgeführt. Die Berechnung der Differenzen wird durch die mit (2) markierte Methode umgesetzt. Für (1) wird für alle in C enthaltenen

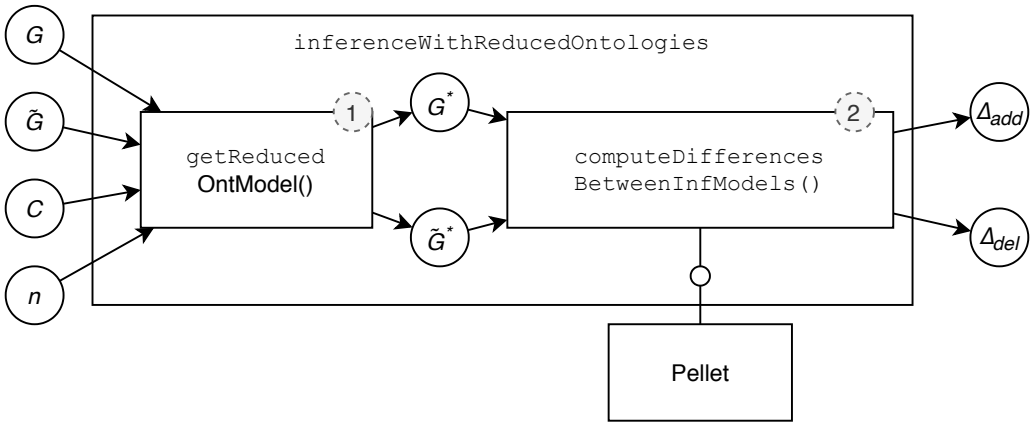

Abbildung 6.4: Die Methode `inferenceWithReducedOntologies`

Knoten die in den Graphen G und $\tilde{G}$ in n Schritten erreichbare Nachbarschaft berechnet. Diese Nachbarschaften werden dann jeweils für die Graphen G und $\tilde{G}$ berechnet. Für G ist das Ergebnis dieser Berechnung entsprechend Definition 5.1.3 $G' = \bigcup_{c \in C} G_{c,n}$ und für $\tilde{G}$ ist das Ergebnis $\tilde{G}' = \bigcup_{c \in C} \tilde{G}_{c,n}$.

G' und $\tilde{G}'$ dienen als Eingabe der Funktion `computeDifferenceBetweenInfModels`. Diese Funktion verwendet einen Open-Source-Fork des OWL Reasoners Pellet [Sir+07], Openllet [Git20], um die Inferenzen von G' und $\tilde{G}'$ zu berechnen. Dieser Reasoner ist in der Lage, die vollständigen Inferenzen der reduzierten Ontologein G' und $\tilde{G}'$ zu berechnen und damit $\Phi_{C,n}(G')$ und $\Phi_{C,n}(\tilde{G}')$, also die Partiellen Inferenzen nach Definition 5.1.4. Die partiellen Inferenzen werden dann verglichen. Die Ausgabe von Pellet ist eine Menge von Statements, die abgeleitet werden können. Es kann in diesen Mengen also analog zu Definition 5.1.5 nach Statements gesucht werden, die nur in einer der Mengen vorkommen. Das Ergebnis dieser Suche bilden die Mengen Δ_{add} und Δ_{del}. Dabei spielt Refactoring im Graph keine Rolle, da Refactoring die Semantik nicht verändern sollte und aus zwei semantisch äquivalenten Graphen die gleichen Inferenzen abgeleitet werden können, also können Unterschiede einfach durch den Unterschied der Mengen betrachtet werden. Dementsprechend werden Refactorings in Δ_{add} und Δ_{del} nicht auftauchen.

Neben der Methode `inferenceWithReducedOntologies` wurde auch eine Methode implementiert, die die Inferenzen auf der gesamten Ontologie berechnet. Diese Methode überspringt einfach den in Abbildung 6.4 mit (1) markierten Schritt und leitet die Daten direkt weiter. Diese Methode wird mit `inferenceWithCompleteOntologies` bezeichnet.

Ein passender Wert für n wurde, wie in Abschnitt 5.1 erwähnt, experimentell gewählt. Dazu wurde eine verhältnismäßig kleine Ontologie verwendet, damit es möglich ist alle Inferenzen zu berechnen. Diese Ontologie sollte aber trotzdem repräsentativ für andere Ontologien mit Ali-

gnments sein. Aus diesem Grund wurde die Mouse Anatomy Ontology, die auch von der OAEI verwendet wird, als Grundlage der Untersuchungen verwendet. Diese Ontologie ist relativ klein und enthält nur 2758 Klassen, dem entsprechend können die Inferenzen für die vollständige Ontologie berechnet werden. Um einen Wert für n zu bestimmen, wurde die Ontologie an zufälligen Stellen verändert. Dann wurden die Ausgaben von `inferenceWithReducedOntologies` und `inferenceWithCompleteOntologies` verglichen, wobei n variiert wurde. Bei $n = 2$ gab es zwischen den Ausgaben der beiden Methoden keine Unterschiede mehr, dem entsprechend wurde dieser Wert als passender Wert für n festgelegt.

6.1.3 Reaktion auf Veränderungen

Zur Reaktion auf Veränderungen wurde das Trait `ChangeFixProposer` definiert. Traits sind mit Interfaces in Java vergleichbar. `ChangeFixProposer` definiert eine Schablonenmethode im Sinne des gleichnamigen Entwurfsmustersi [Gam+95], die Algorithmus 1 umsetzt. Die Methode, die dabei von Implementierungen des Traits umgesetzt werden müssen, sind die Methoden α und γ, für diese Methoden werden also nur Signaturen definiert. Um dieses Verfahren anzuwenden muss also das Trait `ChangeFixProposer` implementiert werden.

6.2 Maschinelles Lernen

Dieser Abschnitt beschreibt die Umsetzung der Ansätze, die maschinelles Lernen einsetzen. Insbesondere bei der Aufbereitung der Daten, die für das Training der Ansätze verwendet werden und bei der Evaluation der Ergebnisse bestehen einige Gemeinsamkeiten in der Implementierung zwischen diesen Ansätzen. Aus diesem Grund werden die Implementierungen an dieser Stelle gemeinsam vorgestellt.

Alle diese Implementierungen verwenden die gleiche Grundstruktur. Diese Struktur und die gemeinsam genutzten Software-Komponenten werden in Abschnitt 6.2.1 dargestellt. Darauf folgt die Beschreibung der Umsetzung des Ansatzes, der Embeddings mit gängigen Klassifikationsverfahren kombiniert (Abschnitt 6.2.2). Abschnitt 6.2.3 stellt die Erweiterung dieser Implementierung um ein Klassifikationsmodell vor. Abschließend wird in Abschnitt 6.2.4 beschrieben, wie die Verwendung von Graph Convolutional Networks umgesetzt wurde.

6.2.1 Übersicht

Die Implementierung aller auf maschinellem Lernen basierenden Ansätze in dieser Arbeit ist in drei Verarbeitungsschritte aufgeteilt:

- **Datenaufbereitung:** in diesem Schritt werden die Daten in eine Form gebracht, die mit Verfahren des maschinellen Lernens verarbeitet werden kann. Dabei werden allen Knoten Labels zugewiesen, dem entsprechend wird der Graph selbst im weiteren Verlauf nicht mehr benötigt.

- **Training:** es wird ein Modell erstellt, mit dem Beispiele klassifiziert werden können

- **Evaluation:** das Modell wird verwendet, um bisher ungesehen Beispiele zu klassifizieren und die Klassifikation dieser bisher nicht betrachteten Beispiele wird mit den gewünschten Ergebnissen verglichen.

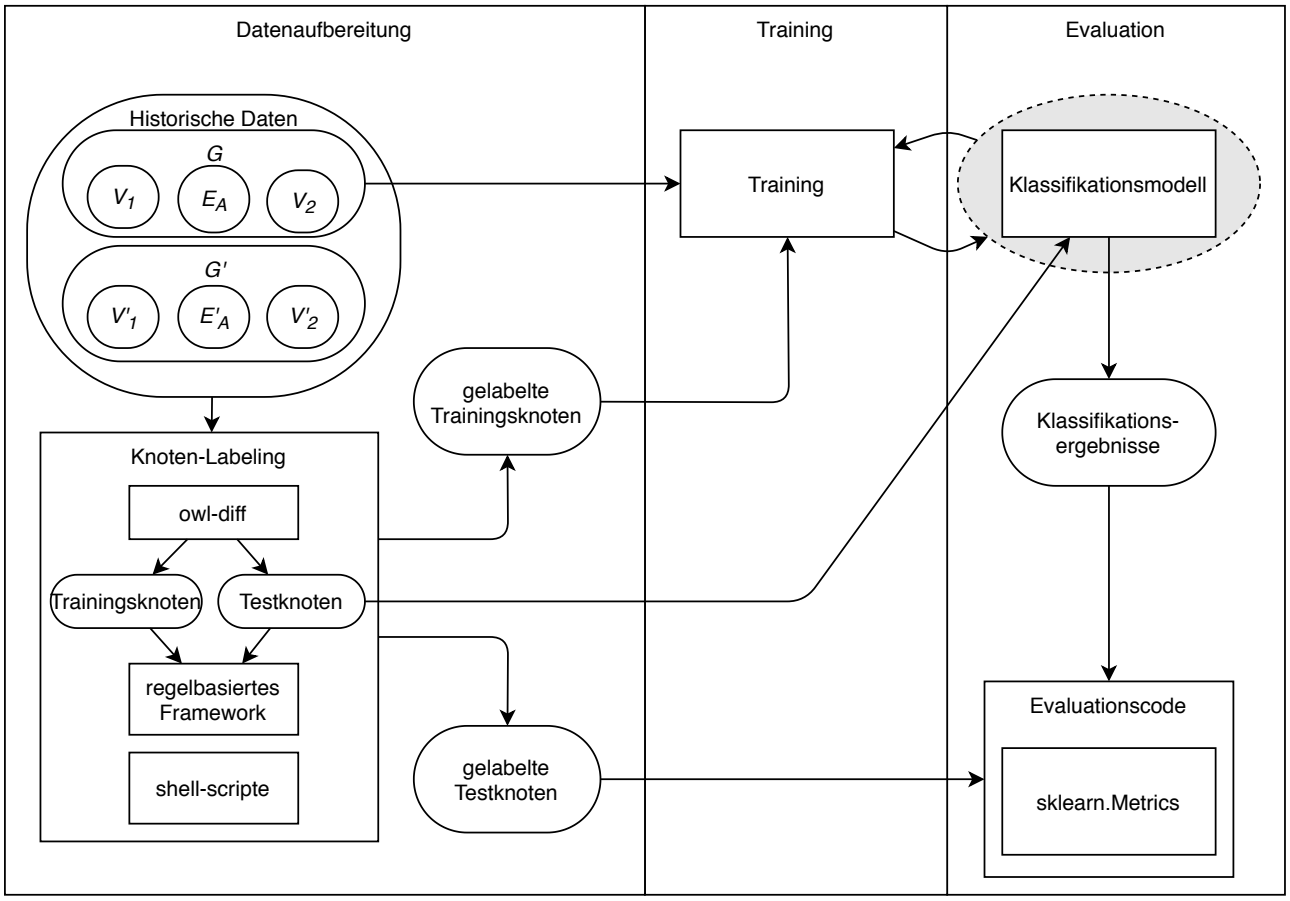

Abbildung 6.5: Übersicht der Implementierung von Verfahren des maschinellen Lernens

In Abbildung 6.5 ist eine Übersicht über diesen Prozess dargestellt inklusive der Komponenten, die alle in dieser Arbeit verwendeten Verfahren des maschinellen Lernens gemeinsam verwenden. Der Prozess beginnt mit historischen Graph-Daten. In der Abbildung sind zwei Versionen eines Wissensgraphen abgebildet, G und G'. Diese Daten werden passend zur Definition 5.2.1 gelabelt. Dazu wird das in Abschnitt 6.1.1 beschriebene Plugin owl-diff in Kombination

mit dem in Abschnitt 6.1 beschriebenen Framework und einige Shell-Scripte verwendet. Als Ergebnis entsteht als erster Zwischenschritt eine Menge von Knoten, die beim Training und beim Test klassifiziert werden sollen und als zweites Ergebnis eine Zuordnung von Labels zu diesen Knoten.

Die Zuordnung wird verwendet, um im Trainings-Schritt ein Klassifikationsmodell, also die Funktion γ, zu erzeugen. Dieses Klassifikationsmodell klassifiziert dann die Testknoten ohne die korrekten Label zu kennen, was eine Zuordnung von Knoten zu Labeln erzeugt. Diese Ergebnisse werden dann mit den aus den historischen Daten berechneten Labeln anhand von verschiedenen Metriken verglichen. Zur Implementierung der Metriken wird das Python-Framework sklearn [Ped+11] eingesetzt. Die genutzten Metriken sind Precision, Recall, Accuracy, F1-Score, ROC-AUC und Average Precision, also genau die in Abschnitt 2.3.3 beschriebene Metriken.

6.2.2 Embedding-Basierte Klassifikation

Die Implementierung des in Abschnitt 5.2 vorgestellten Ansatzes detailliert den im vorigen Abschnitt dargestellten allgemeinen Rahmen und verfeinert diesen an mehreren Stellen. Eine Übersicht ist in Abbildung 6.6 dargestellt. Neue Aspekte im Vergleich zu Abbildung 6.5 sind farblich markiert. Details aus Abbildung 6.5, die nicht verändert wurden, werden an dieser Stelle nicht erneut behandelt. Die zentralen neu hinzugekommenen Komponenten beziehen sich auf die Klassifikation. In der Klassifikation werden zunächst Embeddings erstellt. Dazu wird einerseits das Framework OpenKE [Han+18] verwendet und andererseits die öffentlich verfügbare Implementierung der Autoren von [RP16]. Um diese Frameworks verwenden zu können, müssen die Graph-Daten zuerst noch in ein jeweils passendes Format konvertiert werden, diese wird in der Graph-Konvertierung umgesetzt. Das ist eine Ergänzung des allgemeinen Rahmens. OpenKE verwendet zur Repräsentierung von Wissensgraphen ein proprietäres Format, in dem Graphen in Form von zwei Textdateien repräsentiert werden. Dabei dient eine Textdatei der Zuordnung von Knoten zu Identifiern und eine Textdatei beschreibt in Form von Tripeln, in welchen Relationen die durch diese Identifier repräsentierten Knoten zueinander stehen. RDF2Vec verwendet eine Jena-Datenbank [Apa20] als Eingabeformat.

Die Embedding-Verfahren erzeugen jeweils Trainings- und Testknoten-Embeddings. Mithilfe der Trainingsknoten-Embeddings und der zugehörigen Labels wird im Modul Klassifikationsverfahren ein Modell erzeugt, dass das Trainingsmodul aus dem allgemeinen Rahmen erweitert. Dazu werden verschiedene Klassifikatoren des Python-Frameworks sklearn [Ped+11] eingesetzt. Zu den eingesetzten Klassifikatoren zählen die Klassen `LogisticRegression`, `DecisionTreeClassifier`, `KNeighborsClassifier`, `GaussianNB`, `SVC`, `RandomForestClassifier` und `MLPClassifier`. Die durch diese Klassen erzeugten Klassifikationsmodelle werden dann weiter

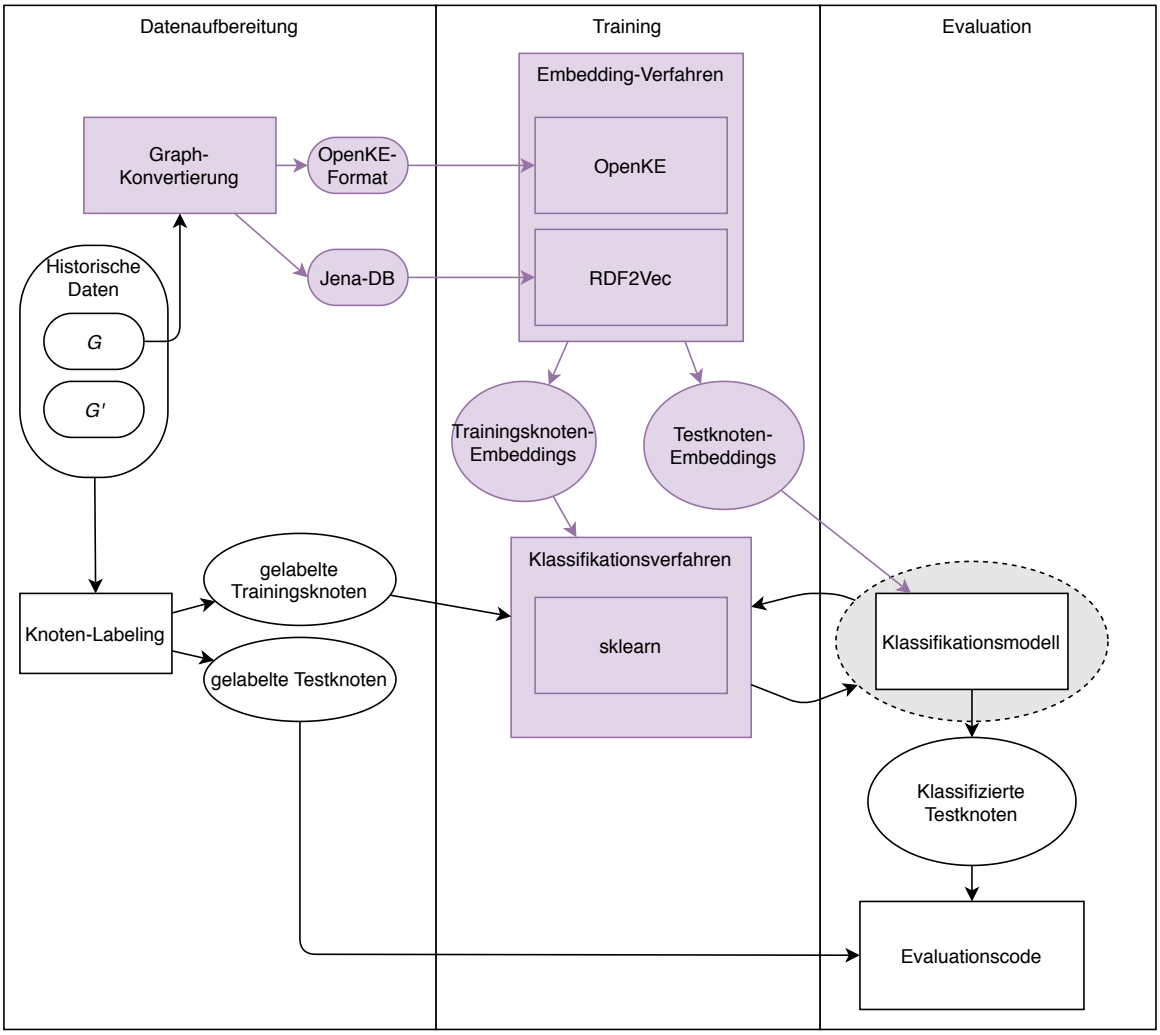

Abbildung 6.6: Implementierung des Embedding-basierten Ansatzes

verwendet wie im vorherigen Abschnitt beschrieben.

6.2.3 Kombination aus Veränderungsmodell und Embeddings

Die im vorigen Abschnitt beschriebene Implementierung wurde erweitert, um den Ansatz zur Kombination von Embeddings und Veränderungsmodell (siehe Abschnitt 5.3) umzusetzen. Eine entsprechende Übersicht ist in Abbildung 6.7 dargestellt. Unterschiede zu der Abbildung 6.6, die den Ansatz darstellt, der nur Embeddings verwendet, sind farbig markiert. Die zentrale Neuerung ist, dass owl-diff in Verbindung mit python-Scripten verwendet wird, um den Knoten im Datensatz eine Veränderungsklassifizierung entsprechend der von owl-diff dargestellten Veränderungsklassen (Also *Added*, *Deleted*, *Annotation Changed* und *Superclass Changed*) zuzuweisen. Das stellt eine Erweiterung des allgemeinen Rahmens dar. Diese Veränderungsklassen werden dann als Vektoren repräsentiert und mit dem Embedding des jeweiligen Knoten

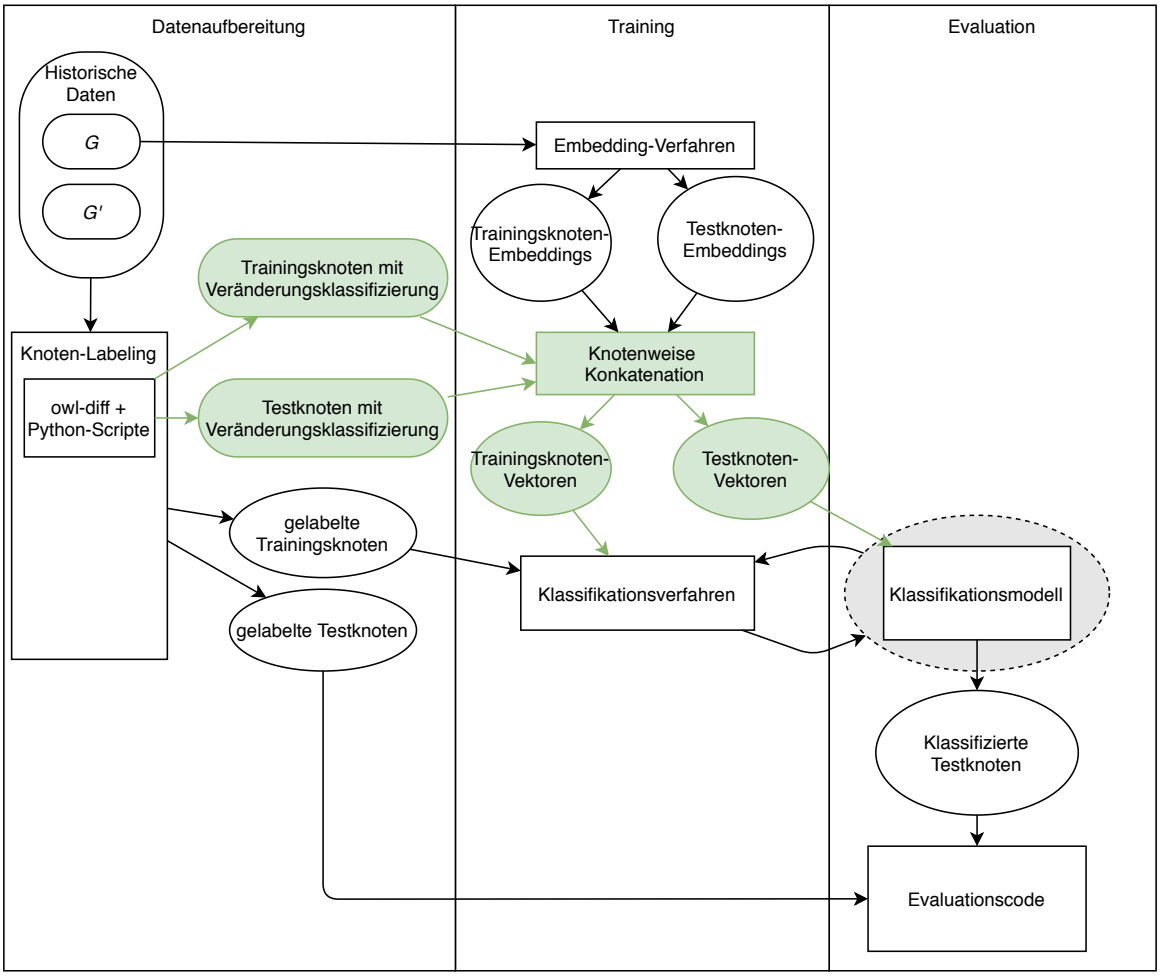

Abbildung 6.7: Implementierung der Kombination aus Embeddings und Veränderungsmodell

konkateniert. Die so entstandenen Vektoren dienen dann als Eingabe für den weiteren Klassifikationsprozess, der genau so funktioniert wie im vorherigen Abschnitt beschrieben.

6.2.4 Graph Convolutional Networks

Auch die Umsetzung der Klassifikation auf Basis von Graph Convolutional Networks basiert auf der gleichen Architektur wie die Ansätze, die Embeddings verwenden. Die Architektur für diese Implementierung ist in Abbildung 6.8 dargestellt. Bei der Implementierung wurde teilweise der Code der Autoren von [Sch+18] verwendet und erweitert. Die Aspekte die durch die ursprüngliche Implementierung von Relational Graph Convolutional Networks umgesetzt werden sind in blau, die Erweiterung in orange dargestellt.

Die frei verfügbare Implementierung von Relational Graph Convolutional Networks musste in der Datenaufbereitung und der Evaluation erweitert werden. Das in python geschriebene RGCN-Modul benötigt für jeden Datensatz, der betrachtet werden soll, ein wenig Implemen-

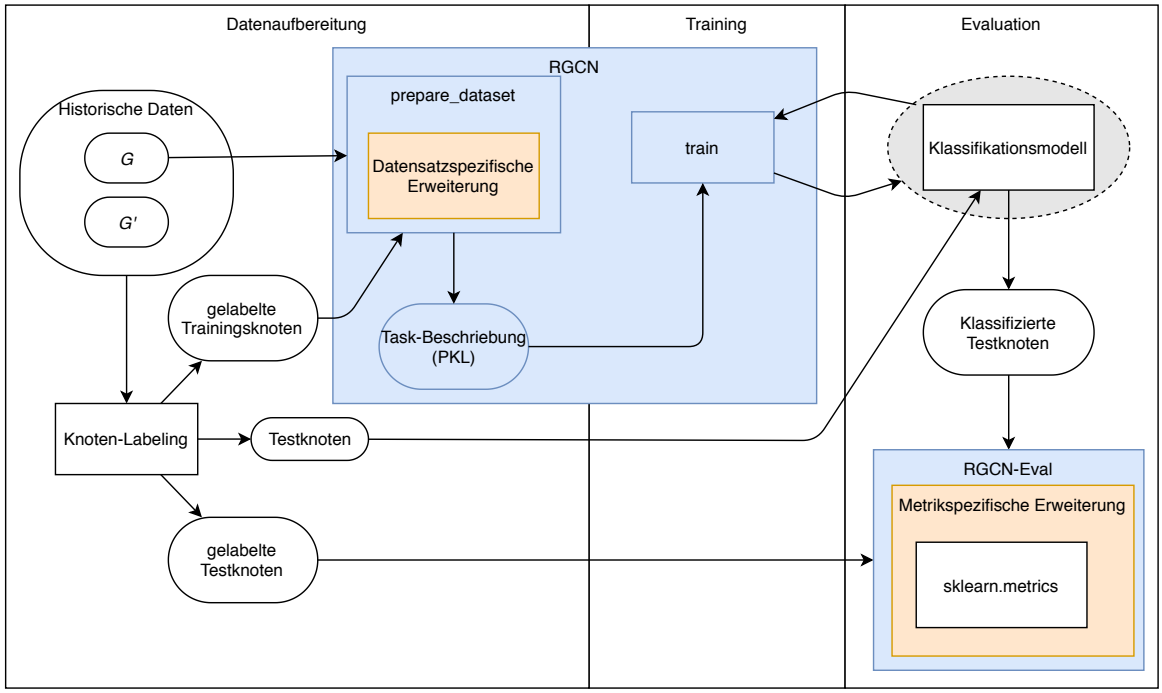

Abbildung 6.8: Implementierung der Klassifikation mit Graph Convolutional Networks

tierungscode, der festlegt, wie der Datensatz inklusive der Trainingsdaten in ein Pickle-File abgelegt werden soll. Pickle [Pyt20] ist eine python-spezifische Form der Datenserialisierung. Außerdem unterstützt die frei verfügbare Implementierung von Graph Convolutional Networks nur die Metriken Accuracy und Precision. Aus diesem Grund wurde die Evaluation um zusätzliche Metriken erweitert, so dass auch die Metriken verwendet werden können, die bei den anderen Ansätzen zur Verfügung stehen.

6.3 Verwendung der Implementierung

In diesem Kapitel wurden zwei Implementierungen vorgestellt: ein Framework, dass die regelbasierte Reaktion auf Veränderungen in Ontologien auf Basis von Veränderungen der Inferenzen umsetzt und ein Rahmen für die Umsetzung von Algorithmen des maschinellen Lernens zur Vorhersage von Veränderungen an Alignments bei gegebenen Ontologie-Veränderungen. Im Zusammenhang mit dieser Umsetzung wurde auch eine Erweiterung des Protégé-Plugins owl-diff vorgestellt.

Im nächsten Kapitel werden diese Umsetzungen verwendet, um die Ansätze zu evaluieren. Dazu wird das Framework zur Einbeziehung von Inferenzen verwendet um ein Anwendungsbeispiel aus dem Bereich der Interweaving Systems umzusetzen. Die Umsetzungen der auf maschinellem Lernen basierenden Ansätze werden anhand eines Datensatzes aus dem biomedizinischen

Bereich evaluiert.

Kapitel 7

Evaluation

Dieses Kapitel beschreibt die Evaluation der in Kapitel 5 vorgestellten Ansätze unter Verwendung der Implementierung, die in Kapitel 6 beschrieben ist. Die verschiedenen Ansätze werden dabei an Beispielen evaluiert, die für die Ansätze als Proof of Concept dienen sollen. Es soll also mit möglichst praxisnahen Beispielen gezeigt werden, dass die Ansätze umsetzbar sind und es soll untersucht werden, wie die Ansätze eingesetzt werden können, was Schwächen der Ansätze sind und welche Vorteile sie mit sich bringen.

Zunächst wird dazu in Abschnitt 7.1 beschrieben, nach welchen Kriterien die Evaluation durchgeführt wird und wie die Anwendungsbeispiele für die Evaluation ausgewählt werden. Der regelbasierte Ansatz zum Einbeziehen von Inferenzen (siehe Abschnitt 5.1) wird anhand eines Beispiels aus der autonomen Verkehrsregelung evaluiert, während die Ansätze, die auf maschinellem Lernen aufbauen (siehe Abschnitte 5.2 und 5.4) auf der Veränderungshistorie mehrerer vernetzter Ontologien evaluiert werden. Die Evaluation des regelbasierten Ansatzes wird in Abschnitt 7.2 vorgestellt. Abschnitt 7.3 beschreibt die Evaluation der anderen Ansätze anhand eines Anwendungsfalls aus der biomedizinischen Informatik.

7.1 Methodik der Evaluation

Im folgenden wird die grundsätzliche Herangehensweise an die Evaluation der in Kapitel 5 vorgestellten Ansätze diskutiert. Dazu werden für die jeweiligen Ansätze zunächst die Voraussetzungen an die Evaluation vorgestellt. Dann wird beschrieben, welche Verfahren zur Beurteilung geeignet sind und was sich daraus für Anforderungen an den jeweiligen Anwendungsfall und die Datenbasis ergeben und welche konkreteren Ziele die Evaluation verfolgt.

7.1.1 Methodik der Evaluation des regelbasierten Ansatzes zur Einbeziehung von Inferenzen

Die Voraussetzungen für die Anwendung der Ansätze sind sehr unterschiedlich: beim regelbasierten Ansatz zur Einbezehung von Inferenzen wird versucht, die Art der Veränderungen vollständig zu fassen und die Regeln werden von Experten konstruiert. Es wird also versucht, auf alle denkbaren Veränderungen zu reagieren und die Art der Veränderungen müssen vollständig erfasst werden. Dementsprechend muss ein Anwendungsbeispiel gewählt werden, in dem das möglich ist. Für diesen Ansatz ist vor allem interessant, wie viel Aufwand die Erstellung der Regeln erfordert und wie schwierig es ist, ein System zu erstellen, dass diese Regeln auswerten kann. Dieser Aufwand kann aus zwei Perspektiven betrachtet werden:

- **Aufwand zum Erstellen der Regeln**: Hier stellt sich die Frage, wie schwierig es ist, passende Regeln zu finden. Außerdem ist zu betrachten, wie effizient diese Regeln notiert werden können.

- **Aufwand zur Erstellung eines geeigneten Systems zur Regelauswertung**: Hier muss untersucht werden, wie viel Ressourcen aufgewendet werden müssen, um ein System zur Auswertung von Regeln zu erzeugen. Dazu muss betrachtet werden, wie viel Aufwand in die Erstellung von abstrakten Modellen fließt. Zu diesen abstrakten Modellen zählen sowohl Verhaltensmodelle als auch Strukturmodelle. Außerdem muss festgestellt werden, wie viel Aufwand zur technischen Umsetzung dieser Modelle benötigt wird. Zusätzlich muss der Rechenaufwand der Umsetzung betrachtet werden.

Es lassen sich für konkrete Anwendungsfälle, bei denen alle Arten von Veränderungen und gültige Reaktionen auf diese bekannt sind, Regeln beschreiben, die diese Veränderungen vollständig erfassen können. Da es bei einem vollständigen Blick auf die Veränderungen relativ einfach möglich ist, die Regeln so zu gestalten, dass sie auf alle Veränderungen adäquat reagieren, ist das Ziel hier dies auch umzusetzen. Sollte dies nicht möglich sein, muss auf Metriken zur Beurteilung der Ergebnisse in realistischen Anwendungsszenarien zurückgegriffen werden, dies ist allerdings in der Evaluation des regelbasierten Ansatzes nicht der Fall.

Um das Finden von Regeln für Veränderungen in den Inferenzen zu erleichtern wäre es ideal, eine Ontologie zu erstellen, an der gezeigt werden kann, dass eine Problemstellung mit dieser Methode elegant gelöst werden kann. Dieser Anwendungsfall muss die folgenden Bedingungen erfüllen: Es werden mindestens zwei konzeptuelle Modelle benötigt, die miteinander verbunden sind. Der durch die Ontologien beschriebene Sachverhalt muss von Veränderungen betroffen sein, die auch Auswirkungen auf das Alignment haben können.

Um die theoretischen und praktischen Grenzen des Ansatzes zu untersuchen wird ein Framework zur Unterstützung der Kommunikation zwischen technischen Systemen, bei denen zum Entwurfszeitpunkt die Kommunikation miteinander nicht berücksichtigt wurde (Interweraving Systems), vorgestellt. An diesem Beispiel soll gezeigt werden, dass der regelbasierte Ansatz in diesem Bereich angewendet werden kann und untersucht werden, welche Stärken und Schwächen der Ansatz hat.

7.1.2 Methodik der Evaluation der auf maschinellem Lernen basierenden Ansätze

Bei den Ansätzen, die maschinelles Lernen verwenden, wird hingegen keine manuelle Betrachtung von Regeln benötigt, aus diesem Grund können sie auch breiter gefasste Problemfälle betrachten: In vielen Fällen ist es nicht möglich, alle möglichen Veränderungen einer Ontologie zu betrachten, insbesondere, wenn diese Ontologien von Menschen bearbeitet werden. Auf einem solchen Anwendungsfall werden die Ansätze, die maschinelles Lernen einsetzen, angewandt. Auch die Zielsetzungen der Ansätze sind unterschiedlich: Der regelbasierte Ansatz versucht konkrete Anpassungen am Alignment vorzuschlagen, während der Machine-Learning-Ansatz lediglich Aligenment-Statements identifiziert, die geändert werden müssten. Der Aufwand zur Erstellung eines Systems, das maschinelles Lernen einsetzt, ist etwas anderes gestaltet, als im regelbasierten Fall. Hier ist vor allem Speicherverbrauch und Rechenzeit interessant, da ein einmal aufgesetztes System potentiell immer wieder eingesetzt werden kann und der reine softwaretechnische Anteil, wie der Aufwand zur Erstellung eines geeigneten Systems, weniger stark ins Gewicht fällt. Dementsprechend wird der Aufwand zur Erstellung des Systems nicht so detailliert betrachtet wie beim regelbasierten Ansatz. Da beim Einsatz von maschinellem Lernen selten perfekte Lösungen erzielt werden können, spielen bei der Bewertung der Güte der Ergebnisse Metriken eine große Rolle, während eine Beurteilung des regelbasierten Ansatzes mit Metriken weniger sinnvoll erscheint. Zur Beurteilung dieses Verfahrens eignen sich verbreitete Klassifkationsmetriken wie zum Beispiel Precision, Recall und F1-Measure. Welche Metriken tatsächlich eingesetzt werden können ist allerdings stark vom Datensatz abhängig, da zum Beispiel bei unbalancierten Datensätzen Metriken wie F1-Measure oder Precision nicht besonders aussagekräftig sind.

Ein für die Evaluation des Machine-Learning Ansatz geeigneter Datensatz muss die folgenden Anforderungen erfüllen: Es werden zwei Ontologien sowie ein Alignment zwischen diesen beiden Ontologien benötigt. Diese Artefakte müssen alle in mindestens drei Versionen vorliegen, die so zusammenpassen, dass sich für drei Zeitpunkte ein konsistenter Graph ergibt. Der Graph muss in mindestens drei Versionen vorliegen, damit mindestens zwei Versionssprünge betrach-

tet werden können. Einer der Versionssprünge wird zum Trainieren des Modells verwendet, der andere Versionssprung zur Evaluation des Modells.

Dazu wird ein Datensatz aus dem Bereich der Biomedizin betrachtet. Da für den Datensatz Alignments und Ontologien in mehreren Versionen vorliegen, soll anhand des Datensatzes untersucht werden, ob ein Machine-Learning-Ansatz überhaupt angewandt werden kann und wie die verschiedenen Ansätze im Vergleich zueinander abschneiden.

7.2 Regelbasierter Ansatz

In diesem Abschnitt wird der regelbasierte Ansatz anhand eines Beispiels aus der Verkehrsregelung untersucht. Einige der dazu hier vorgestellten Überlegungen wurden bereits in [JI17; JI20] publiziert. Dabei werden folgende Ziele verfolgt:

1. Es sollen die theoretischen und praktischen Grenzen des Ansatzes untersucht werden.
2. Dazu wird ein Ansatz für die Abbildung von Interweaving Systems im allgemeinen vorgestellt. Dieser Ansatz verwendet Ontologien und ist auf die Berechnung von Inferenzen angewiesen.
3. Es soll gezeigt werden, dass mit einem regelbasierten Ansatz zur Einbeziehung von Inferenzen ein Problem aus dem Gebiet der Interweaving Systems gelöst werden kann.
4. Außerdem soll untersucht werden, welche Vorteile dieser Ansatz insbesondere aus softwaretechnischer Perspektive mit sich bringt und
5. welche Probleme auftreten können, wenn dieser Ansatz angewandt wird.

Dazu wird zunächst das Anwendungsbeispiel und die innerhalb des Anwendungsbeispiels zu bearbeitende Zielsetzung beschrieben, um einen Eindruck über Interweaving Systems im Allgemeinen und den Anwendungsfall zu erhalten. Anschließend wird, passend zum zweiten Ziel, zunächst ein allgemeiner Ansatz für Interweaving Systems vorgestellt und im weiteren Verlauf der Evaluation betrachtet, wie dieser allgemeine Ansatz verwendet werden kann, um eine Problemstellung aus dem Bereich der autonomen Ampelsysteme zu betrachten. Dieser Ansatz wurde so gewählt, dass er die in Abschnitt 7.1 vorgestellten Anforderungen erfüllt.

Interwoven Systems [Tom+14] bezeichnet die nicht-hierarchische Verflechtung von Systemen, die nicht dafür ausgelegt wurden, miteinander zu interagieren. Dies betrifft sowohl die direkte Interaktion (wie Kommunikation) zwischen den Systemen, als auch die indirekte Beeinflussung durch Veränderungen des Systemumfelds durch andere Systeme. Diese Verflechtung

stellt insbesondere die Systementwickler vor Herausforderungen, da nicht alle wichtigen Modelle und Datenquellen zur Entwurfszeit bekannt sein können. Zur Lösung dieses Problems können die Verwendung von Ontologien und Verbindungen durch Ontolgie-Alignments einen wichtigen Beitrag leisten. Ein Modell zur Lösung eines solchen Problems wird in dieser Arbeit entworfen, um den regelbasierten Ansatz zu evaluieren. Aus diesem Grund eignet sich der Anwendungsfall für eine Evaluation des regelbasierten Ansatzes. Der Ansatz, der für den Anwendungsfsall vorgestellt wird, ist zwar nicht auf Interweaving Systems beschränkt, wird aber nur für diesen Bereich betrachtet, um die Grenzen des regelbasierten Ansatzes in einem konkreten und realistischen Anwendungsszenario zu untersuchen.

7.2.1 Anwendungsfall: Autonome Ampelsteuerung

Um einen Eindruck von Interweaving Systems zu erhalten, wird zuerst der Anwendungsfall vorgestellt, anhand dessen der Ansatz für Interweaving Systems demonstriert wird: Autonome Ampelsysteme können anhand der ihnen vorliegenden Daten selbst entscheiden, wie lang bestimmte Ampelphasen sein sollen, um so den Verkehrsfluss zu optimieren. Typische Systeme aus diesem Bereich wie [Ste+16] betrachten nur einzelne Ampelsysteme. Der Austausch von Informationen ist meist nicht vorgesehen. Es kann allerdings sinnvoll sein, auch Verkehrsdaten von anderen Systemen mit einzubeziehen, insbesondere in extremen Verkehrssituationen an anderen Kreuzungen, wie Staus und Straßensperrungen – es kann sich lohnen, zu diesen besonderen Verkehrsereignissen Daten verwenden zu können, bevor sich die Auswirkungen an der lokalen Kreuzung im Verkehrsfluss zeigen.

Es gibt zwar Ansätze zu autonomen Systemen, die Informationen austauschen um zum Beispiel eine gemeinsame Ampelphase entstehen zu lassen [Pro+09], allerdings werden strukturelle Veränderungen in diesen Ansätzen nicht untersucht. Wenn Ontologien in diesem Bereich eingesetzt werden, werden sie häufig als eine Repräsentation der Daten verwendet. Dies ist zum Beispiel in [Fer+16] der Fall: es wird eine Ontologie-Schicht als Repräsentation von verschiedenen Verkehrssituationen verwendet. In dem in diesem Abschnitt vorgestellten Anwendungsfall werden autonome Ampelsteuerungssysteme betrachtet, die Ontologien verwenden, um ihre Umgebung darzustellen. Veränderungen in der Umgebung werden als Veränderungen in den Ontologien dargestellt. Es werden auch strukturelle Veränderungen betrachtet. Anhand dieses Anwendungsfalls soll der regelbasierte Ansatz untersucht werden.

Der Kern des in diesem Abschnitt vorgestellten Anwendungsfalls ist ein Ampelmanagementsystem, welches Verkehrsflussdaten verwendet, um die Schaltpläne der Ampeln an die aktuelle Verkehrssitutation anzupassen. An jeder Kreuzung werden Daten zum Verkehrsfluss gemessen. Die Steuerungssysteme der Kreuzungen können auch auf Messdaten von anderen Kreuzungen

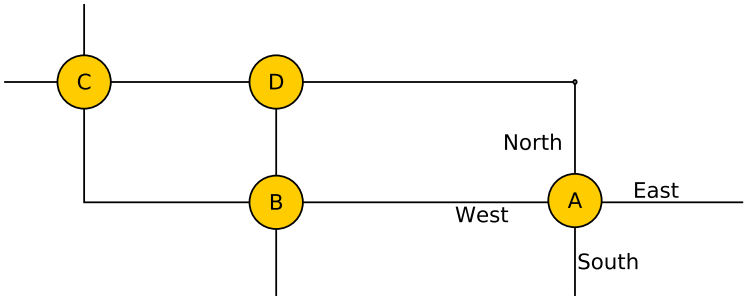

Abbildung 7.1: Kreuzungsnetz als Anwendungsbeispiel

zugreifen. Dabei soll das System gegenüber Veränderungen in der Umwelt robust sein, auch wenn diese Veränderungen struktureller Natur sind. Das System soll also weiter den Informationsaustausch unterstützen können, wenn Veränderungen auftreten. Das verwendete Modell betrachtet die folgenden Veränderungen:

- *Abschaltung und Hinzukommen von Ameplsteuerungssystemen.* Es kann vorkommen, dass Ampelsteuerungen ausgeschaltet werden oder ausfallen. Genauso kann es vorkommen, dass ein abgeschaltetes Ampelsteuerungssystem wieder in Betrieb genommen wird oder ein neues System hinzukommt. Ebenso wird Abschaltung und Einschaltung von einzelnen Sensoren zur Messung des Verkehrsflusses betrachtet.

- *Staus sowie die vollständige Sperrung von Straßen.* Straßen können durch Staus oder Sperrungen wegen Bauarbeiten vollständig blockiert werden. Diese Blockaden können auch wieder aufgelöst werden.

Das System in dem hier vorgestellten Anwendungsfall soll auf diese Veränderungen selbstständig und autonom reagieren können. Als Beispiel zur Erläuterung des Ansatzes wird das in Abbildung 7.1 dargestellte Netz aus Straßenkreuzungen verwendet. Im Gegensatz zu detaillierteren Darstellungen in der Literatur zu Verkehrsplanung (zum Beispiel in [Shi+19]) wird eine abstraktere Notation zur einfachen Verständlichkeit verwendet, was allerdings keinen Einfluss auf die allgemeine Anwendbarkeit des vorgestellten Ansatzes hat: Jede Kreuzung wird mit einem Buchstaben bezeichnet. Die Straßen, die diese Kreuzungen Verbinden werden als Linien zwischen diesen Kreuzungen repräsentiert. Verschiedene Ein- und Ausfahrten werden mit der jeweiligen Himmelsrichtung (North/East/South/West) bezeichnet.

In dem hier dargestellten Modell werden Verkehrsflussdaten für jede Straße gemessen, die in eine Kreuzung hinein und aus der Kreuzung hinaus führt. Mögliche Metriken zur Beobachtung des Verkehrsflusses, die gemessen werden können, sind unter anderem die *Verkehrsdichte*, die *Verkehrsstärke* und die *Zeitlücke* [For12]. Die *Verkehrsdichte* ist die Anzahl der Fahrzeuge, die sich zu einem Zeitpunkt auf einem bestimmten Streckenabschnitt befinden. Die *Verkehrsstärke* ist die Anzahl der Fahrzeuge, die einen Straßenquerschnitt pro Zeiteinheit durchfahren. Die *Zeitlücke* ist die Zeit, die zwischen dem Verlassen eines Querschnitts und dem Betreten eines Querschnitts des darauffolgenden Fahrzeugs vergeht. Diese Daten können mit Steuerungssystemen an anderen Kreuzungen ausgetauscht werden. Von welchen anderen Kreuzungen Messdaten betrachtet werden sollen, entscheiden die Steuerungssysteme ebenfalls selbst.

Technisch können die Messdaten zum Beispiel über ein Publish-Subscribe/Blackboard-Framework kommuniziert werden, einzelne Ampelsysteme könnten dann entscheiden, welche Messdaten sie abbonieren würden. In dem Blackboard-System können auch Korrelationsscores zwischen Sensorwerten berechnet werden. Korrelationsscores werden in dem hier diskutierten Beispiel genutzt, um beim Ausfall eines Sensoren einen Ersatz zu finden.

Steuerungssysteme an den Kreuzungen verfügen über verschiedene Schaltpläne und können regelbasiert entscheiden, welcher Schaltplan verwendet werden soll. Eingabe für die Regeln sind Verkehrsflussdaten, diese können sowohl lokal als auch an anderen Kreuzungen gemessen werden. Der hier verwendete Ontologiebasierte Ansatz soll es den Ampelsteuerungen erlauben, lokal zu entscheiden, welche Schaltpläne verwendet werden und trotzdem Informationen von anderen Ampelsteuerungssystemen in diese Entscheidung mit einzubeziehen.

Im folgenden Abschnitt wird ein allgemeiner, auf Ontologien basierender Ansatz für Interweaving Systems im allgemeinen vorgestellt. Dieser Ansatz ist von dem in diesem Abschnitt vorgestellten Ansatz unabhängig und wird später auf den Anwendungsfall angewandt.

7.2.2 Ontologiebasierter Ansatz für Interweaving Systems

Die Kernidee des in [JI20] vorgestellten Ansatzes zur Unterstützung von Interweaving Systems ist es, ein gemeinsam verwendetes Modell der Umgebung zu erzeugen, welches repräsentiert, welche Informationen alle beteiligten Systeme über ihre Umwelt erlangen können. Dieses Modell wird als *Directory Model* bezeichnet. Ein weiteres, für jedes beteiligte System vorhandene *Connector Model* wird dann verwendet um zu beschreiben, welche Informationen direkt und indirekt aus anderen Systemen verwendet werden sollen. Es wird also verwendet, um zu beschreiben, welche Daten ausgetauscht werden. Die Implementierung dieser Strategie basiert auf einem domänenunabhängigen Framework, welches Ontologien und andere semantische Technologien verwendet.

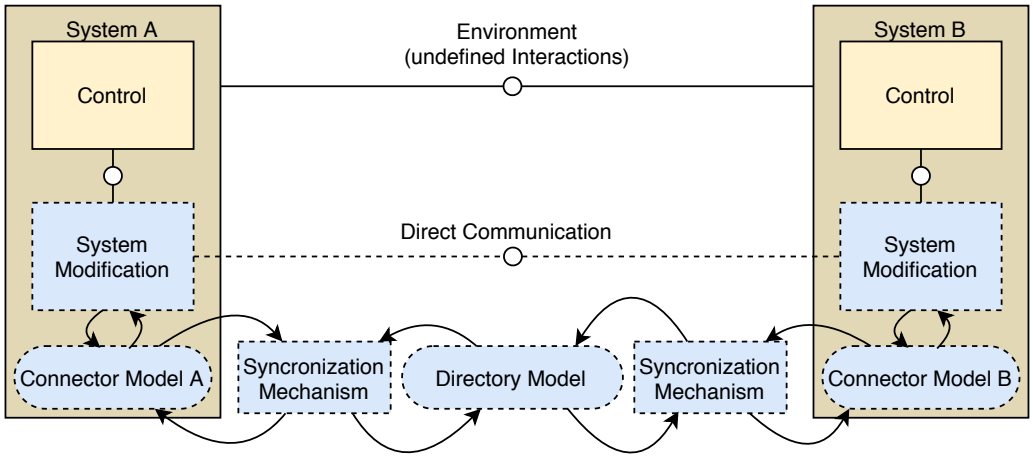

Abbildung 7.2: Ansatz zur Erweiterung der Interweaving Systems

Ein vereinfachtes Beispiel für die an einer Anwendung eines solchen Ansatzes beteiligten Komponenten wird in Abbildung 7.2 dargestellt. In der Abbildung werden Ontologien der Einfachheit halber als Speicher repräsentiert, sie können allerdings auch als eine Sicht auf Konzepte realisiert werden, die in mehreren Speichern vorgehalten werden. Typische, bereits existierende Komponenten von Interweaving Systems sind in gelb dargestellt. Die Erweiterungen der Interweaving Systems sind in blau dargestellt.

Die verwendeten Modelle müssen vor allem zwei Aufgaben erfüllen: (1) die Systeme dabei unterstützen herauszufinden, welche Informationen über die Umgebung verfügbar sind und (2) Informationen über die Umgebung, die andere Systeme beobachten können, mit lokalen Informationsbedürfnissen verbinden. Die erste Aufgabe wird mit dem als *Directory Model* bezeichneten Modell umgesetzt. Dieses Modell repräsentiert, welche Systeme in der Lage sind Informationen auszutauschen und welche mit den Systemen verbundenen Sensoren dazu zur Verfügung stehen.

Die zweite Aufgabe wird mit dem als *Connector Model* bezeichneten Modell umgesetzt. Diese Umsetzung des Connector Models, das für jedes System vorhanden ist, repräsentiert direkt und indirekt verbundene Sensoren. Indirekt verbundene Sensoren sind jene Sensoren, die an anderen Systemen angeschlossen sind und für das lokale System relevante Informationen liefern. Im Directory Model werden Sensoren mit Daten-Anforderungen verbunden. Diese Anforderungen werden genutzt, um die Anforderungen der Selbst-Management-Möglichkeiten eines Systems zu repräsentieren. Außerdem enthält das Directory Model ein *Similarity Model*, das Ähnlichkeitsdaten speichert. Diese Daten werden verwendet, um Sensoren anderer Systeme mit möglichen Ersatz-Sensoren zu verbinden, falls ein Sensor oder System ausfallen soll. Dies wird so umgesetzt, dass Paaren von Sensoren ein Wert zugeordnet wird, der ihre Ähnlichkeit zueinander repräsentieren soll.

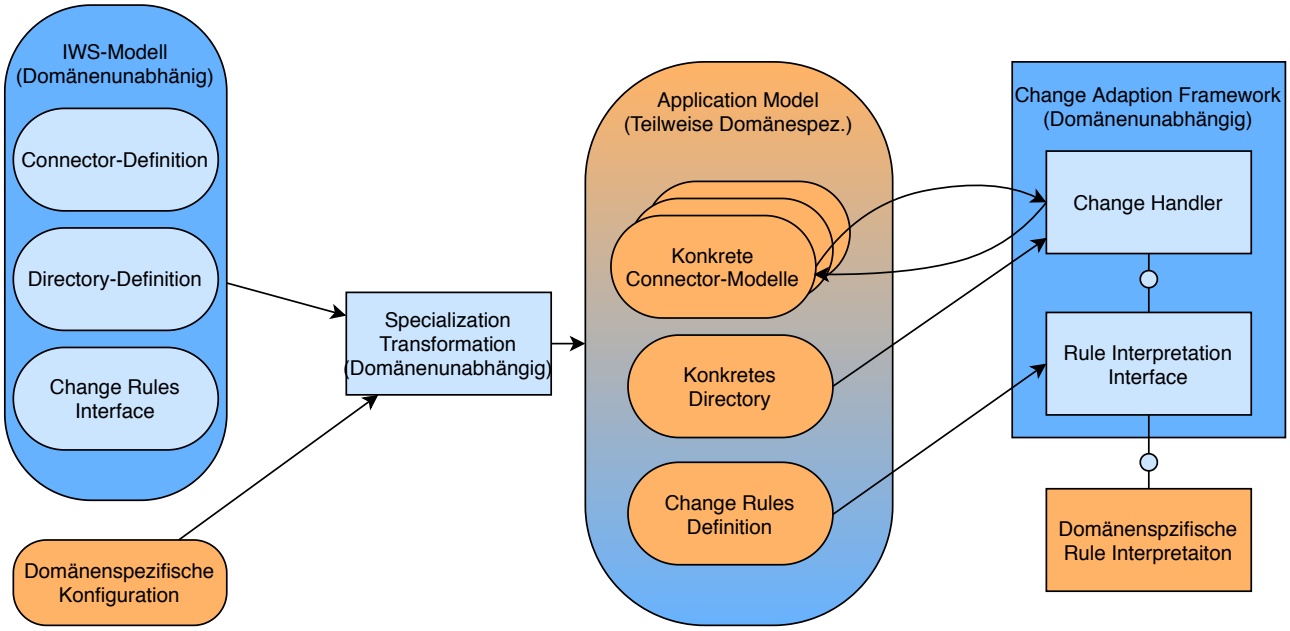

Abbildung 7.3: Übersicht eines Ontologie-basierten Ansatzes: Domänenunabhängiges IWS-Modell wird in teilweise domänenspezifisches Modell umgewandelt.

Ein Synchronisationsmechanismus integriert das Directory Model mit den lokalen Connector-Modellen. Dieser Synchronisationsmechanismus muss außerdem sicherstellen, dass das Directory-Modell stets aktuelle Informationen enthält und jedem System den Zugriff auf das Directory-Modell ermöglichen. Entsprechend kann dieser Mechanismus durch einen regelmäßigen Broadcast aller Systeme umgesetzt werden. Auf Basis der mit dem Broadcast verbreiteten Informationen können die Systeme miteinander interagieren, um Daten auszutauschen. Dieser Datenaustausch kann zum Beispiel durch ein Peer-to-Peer Framework wie DDS [Par03] oder eine andere Topic-orientierte Middleware umgesetzt werden, die nicht auf einen zentralen Broker angewiesen ist. Diese Middleware könnte natürlich auch genutzt werden, um die Daten der Directory Ontology auszutauschen.

Dabei sollten alle Modelle und der Mechanismus zum Umgang mit Veränderungen soweit wie möglich mit standartisierten Ontologiesprachen umgesetzt werden. Das ermöglicht die Verarbeitung dieser Modelle mit standardisierten und frei verfügbaren Tools. Soweit möglich sollte nur Subclass Reasoning eingesetzt werden, welches nur wenige Ressourcen benötigt. Dementsprechend ist Subclass Reasoning insbesondere in Anwendungsszenarien relevant, in denen Rechenzeit oder Speicherplatz stark beschränkt sind, was in Interweaving Systems häufig der Fall ist.

Ein Überblick über die wichtigsten Modelle und Komponenten zur Übertragung des Ansatzes an eine Domäne ist in Abbildung 7.3 dargestellt. Domänenunabhängige Aspekte sind blau, dommänenspezifische Aspekte sind orange abgebildet. Das domänenunabhängige *IWS-Modell*

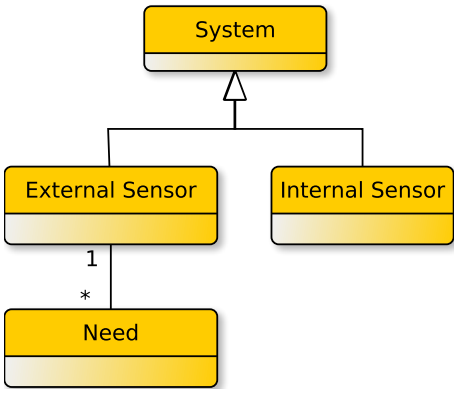

Abbildung 7.4: Beziehung der zentralen Konzepte aus Connector- und Directory Ontology

definiert Konzepte für das Directory- und Connector-Modell sowie eine abstrakte Darstellung von *change rules*. Um diese Modelle in einem domänenspezifischen Anwendungsfall nutzen zu können, wird eine *domänenspezifische Konfiguration* benötigt. Das IWS-Modell und die domänenspezifische Konfiguration werden als Eingabe für die *Specialization Transformation* genutzt. Diese Spezialisierung erweitert das IWS-model mit domänenspezifischen Aspekten und erzeugt das *Application Modell*. Das Application Modell enthält konkrete Connector- und Directory Modelle, die die tatsächlichen domänenspezifischen Systeme und ihre Möglichkeiten zur Beobachtung der Umgebung modellieren. Dieses Modell enthält außerdem *Change Rules*, die zu dem *Change Rule Interface* passen müssen. Diese Modelle erlauben den Systemen miteinander zu kommunizieren und Daten auszutauschen.

Das domänenunabhängige *Change Adaption Framework* wird verwendet, um Fehlertoleranz in dieses Modell einzubringen. Das Framework wird aktiv, wenn im Connector- oder Directory Model Änderungen auftreten. In diesem Fall werden die Change Rule Definitions verwendet, um adäquat auf die Veränderungen zu reagieren. Um eine domänenspezifische Veränderungsreaktion zu unterstützen, gibt es ein Interface für die domänenspezifische Abarbeitung von Regeln. Mithilfe dieser Schnittstelle können die Regeln ausgewertet werden und die Conncetor-Modelle an Veränderungen angepasst werden.

Außerdem sollten der in diesem Abschnitt vorgestellte Ansatz die in Abschnitt 5.1.3 vorgestellten Einschränkungen zur Anwendbarkeit des regelbasierten Ansatzes erfüllen. Einige der Vorbedingungen sind bereits durch die Domäne gefordert (zum Beispiel, dass nur Subclass-Reasoning eingesetzt werden sollte), andere Vorbedingungen beziehen sich in erster Linie auf die Darstellung der Ontologien und müssen dementsprechend beim Entwurf der Ontologien betrachtet werden.

Diese Prinzipien führen zu einer Umsetzungsstrategie für die Directory und Connector Ontology, die jeweils eine ontologiebasierte Umsetzung von Directory- und Connector-Modell dar-

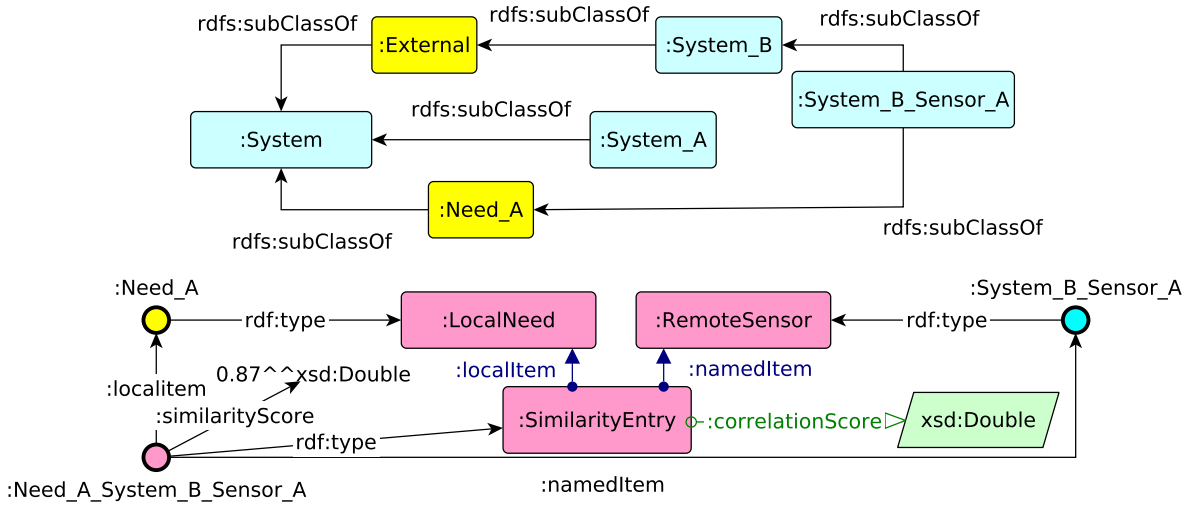

Abbildung 7.5: Beispielhafte Connector Ontology (gelb) mit per Alignment verbundenen Klassen aus der Directory Ontology (blau) inklusive der Similarity Ontology (unten)

stellen. Die zentralen Konzepte dieser Umsetzung sind in Abbildung 7.4 dargestellt. Die Klasse *System* enthält alle Daten von Sensoren. Wenn ein Sensor Teil eines Systems ist, ist die Klasse, die die von dem Sensor gemessenen Daten repräsentiert, eine Unterklasse der Klasse, die das entsprechende System repräsentiert. Das Konzept *Need* markiert für ein lokales System, wenn ein externer Sensor mit eingebunden werden soll.

Ein Beispiel für eine Ontologie-basierte Umsetzung ist in Abbildung 7.5 in einem Graffoo-Diagramm [Fal+14] dargestellt. In Graffoo-Diagrammen werden Klassen durch Rechtecke und Datentypen durch Rhomben repräsentiert. Ein konkreter Sensorwert würde als Instanz in dieser Ontologie abgebildet. Die Connector Ontology repräsentiert das *Need*-Konzept und gibt an, welche Sensoren als *External* markiert werden. Alle Sensoren, die als Unterklasse von *External* definiert werden, sind Sensoren, die an anderen Systemen angeschlossen sind und dem lokalen System Daten senden können. Alle Verbindungen zwischen der Directory Ontology und der Connector Ontology werden durch Alignment-Statements umgesetzt. Dementsprechend kann bei Veränderungen an der Directory- und Connector Ontology das in dieser Arbeit vorgestellte Verfahren zum Anpassen von Ontologie Alignments angewandt werden. Die Similarity Ontology als Teil der lokalen Ontologie ist im unteren Teil der Abbildung 7.5 dargestellt. Diese Teil-Ontologie besteht aus *Similarity Entries*, die Paare aus lokalen Datenanforderungen und an anderen Systemen angeschlossenen Sensoren bildet.

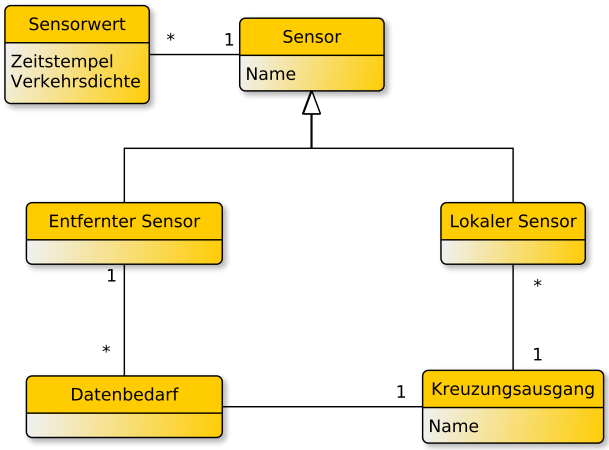

Abbildung 7.6: Konzeptuelles lokales Modell einer Ampelsteuerung

7.2.3 Umsetzung des Anwendungsfall

Die Lösung baut darauf auf, dass jede Ampelsteuerung über ein *lokales Modell* verfügt. In dem lokalen Modell wird beschrieben, welche externen Datenquellen vewendet werden und welche lokalen Datenquellen es gibt. Außerdem gibt es ein *globales Modell*. Das globale Modell beschreibt, welche Datenquellen exisiteren und enthält weitere schematische Informationen. Wie Sensorwerte statistisch zusammenhängen, wird in einem dritten Modell, dem *Korrelationsmodell*, festgehalten. Wenn im Betrieb des Systems Änderungen auftreten, wird regelbasiert auf diese Änderungen reagiert.

Konzeptuelles Modell

Ein UML-Klassendiagramm der Konzepte des lokalen Modells einer Ampelsteuerung ist in Abbildung 7.6 abgebildet. Zur Unterstützung der Auswahl einer Ampelschaltung verwendet die Ampelsteuerung Messdaten, diese werden durch die Klasse *Sensorwert* abgebildet. Jedes Datum ist einem *Sensor* zugeordnet. Jede Ampelsteuerung an einer Kreuzung verwaltet mehrere Kreuzungsausgänge. Diese können mit *lokalen Sensoren* zur Messung der Verkehrsdichte versehen sein. Zusätzlich kann der Datenbedarf an einer Kreuzung durch weitere Sensoren erfüllt werden, die von entfernten Ampelsteuerungssystemen verwaltet werden. Diese Sensoren werden als *Entfernte Sensoren* bezeichnet. Sensoren, die als entfernte Sensoren im lokalen Modell abgebildet werden können, stammen aus dem Globalen Modell.

Die Konzepte des Globalen Modells sind in Abbildung 7.7 dargestellt. Das globale Modell wird als Verzeichnis verwendet, um Datenquellen auswählen zu können. Als Datenquellen stehen *Sensoren* zur Verfügung. Jeder Sensor gehört zu genau einer *Ampelsteuerung*. Beide können jeweils über den *Namen* adressiert werden.

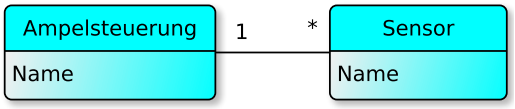

Abbildung 7.7: Konzeptuelles globales Modell

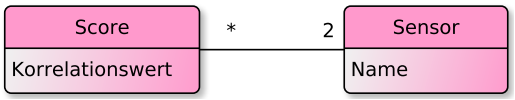

Abbildung 7.8: Konzeptuelles Korellationsmodell

Zur Repräsentierung der Korrelation zwischen einzelnen Sensoren existiert ein Korrelationsmodell. Ein UML-Diagramm der Konzepte dieses Modells ist in Abbildung 7.8 dargestellt. Der *Score* bildet einen Korrelationswert zwischen verschiedenen Sensoren ab. Jeder *Score* ist mit zwei *Sensoren* verbunden.

Diese einfachen Modelle könnten mit einem klassischen modellbasierten Ansatz umgesetzt werden. Die Semantik der Modelle und wie mit ihnen umgegangen werden soll müsste dann durch zusätzlichen Code oder weitere UML-Diagramme (zum Beispiel Aktivitätsdiagramme) definiert werden. In dem hier vorgestellten Beispiel wird die Problemstellung stattdessen damit gelöst, dass eine Ontologie und verschiedene Werkzeuge aus dem Semantic-Web-Umfeld verwendet werden.

Ontologiebasierte Umsetzung

Die im vorigen Abschnitt dargestellten Modellierungen lassen sich sehr direkt in das in Abschnitt 7.2.2 vorgestellte Framework übertragen: Das globale, lokale und Korrelations-Modell werden jeweils in einer Ontologie abgebildet. Dies führt zu einer *Directory Ontology*, die schematische Informationen enthält und so als eine Art Interface-Format bereitstellt, einer *Correlation Ontology*, die beschreibt, wie Datenquellen verbunden sind und einer *Connector Ontology*, die beschreibt, welche Datenquellen von anderen Systemen verwendet werden. Teile der connector ontology können auch als Binding von Variablen betrachtet werden.

In Abbildung 7.9 ist die connector ontology einer Kreuzung A inklusive des Alignments zur directory ontology dargestellt. Klassen der directory ontology sind in blau abgebildet. Die Klasse *Intersection* ist die Oberklasse für alle Messungen die vorgenommen werden. Dementsprechend ist sie mit einem Zeitstempel und einem Sensorwert versehen. Für jedes Ampelsteuerungssystem existiert eine Unterklasse von Intersection. Intersection bildet die Klasse Ampelsteuerung aus dem konzeptuellen Modell ab. Für jede Ampelsteuerung existiert eine Unterklasse von Intersection, in der Abbildung als die Klassen *A, B, C* und *D* bezeichnet. Zugehörige *Sensoren* werden ebenfalls durch Unterklassen dargestellt: Für jeden Sensor existiert eine Unterklasse.

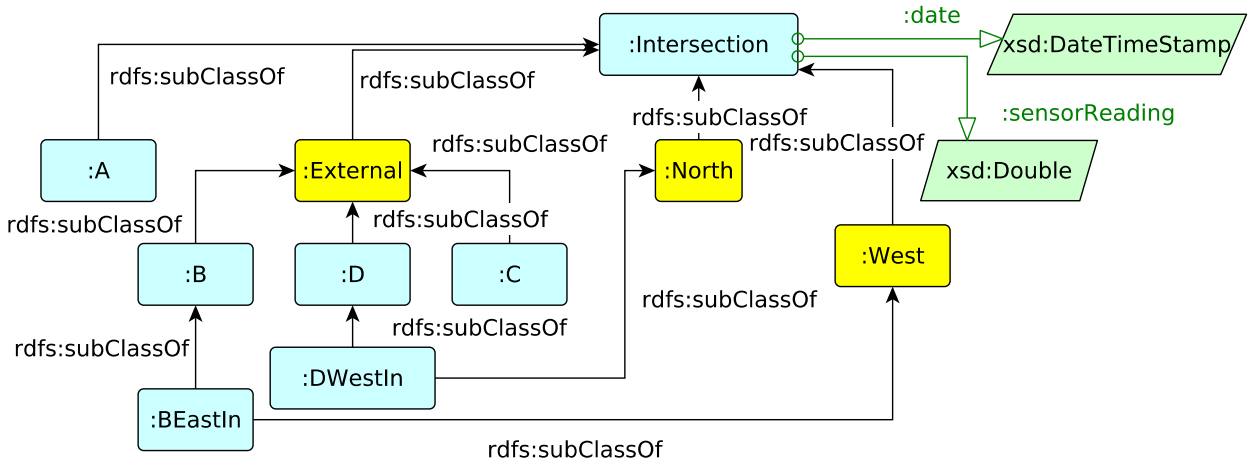

Abbildung 7.9: Lokale Ontologie einer Ampelsteuerung an einer Kreuzung *A* mit Ausschnit der globalen Ontologie

Einzelne Datenpunkte werden als Instanzen der Sensorklassen abgebildet. Unterklassen für konkrete Kreuzungen sind in dem dargestellten Beispiel die Klassen *A, B, C,* und *D*. *BEastIn* und *DWestIn* sind Klassen zur Repräsentation von konkreten Sensoren. Zum Beispiel repräsentiert *BEastIn* alle Sensormessungen, die durch das Ampelsteuerungssystem *B* durch einen bestimmten Sensor, nämlich durch den als „BEastIn“ bezeichneten Sensot, gemessen wurden. Diese Klassen können auch automatisiert durch Software generiert werden. In der Praxis ist das eine geeignete Herangehensweise.

Klassen der connector ontology werden in gelb dargestellt. Die Klasse *External* markiert alle Messungen, die nicht in dem lokalen Steuerungssystem selbst anfallen. Da die dargestellte Ontologie das Steuerungssystem *A* repräsentiert, sind alle Datenpunkte, die nicht eine Unterklasse von *A* sind als *External* markiert. Die Klassen *North* und *West* stellen einen Bedarf für externe Daten dar. Durch die Unterklassenbeziehung zwischen *BEastIn* und *West* wird dargestellt, dass der Bedarf für externe Daten befriedigt ist. Die Beziehung zwischen angeforderten Daten und den Klassen, die die passenden Messungen repräsentieren, betrifft Klassen aus der lokalen und der globalen Ontologie. Die Beziehung ist also ein Alignment.

Das Alignment wird auf Basis von statistischem Wissen über den Zusammenhang zwischen Messwerten angepasst. Der Zusammenhang zwischen den Messwerten soll repräsentieren, wie gut Messwerte von einem Sensor geeignet sind, Messwerte des anderen Sensors zu ersetzen. Wie genau dieser Zusammenhang berechnet wird ist außerhalb des Fokus dieser Arbeit und wird dementsprechend nicht betrachtet. Ein konzeptuelles Modell mit beispielhaften Werten ist in Abbildung 7.10 dargestellt. Elemente der lokalen Ontologie werden in gelb, Elemente der globalen Ontologie in blau und Elemente der Korrelationsontologie in pink dargestellt. Die

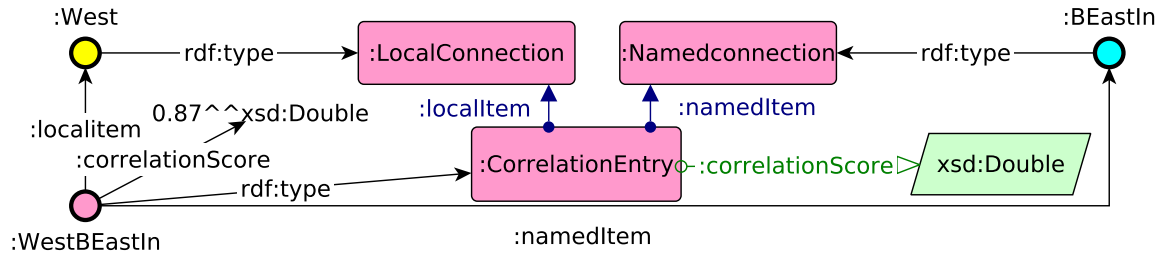

Abbildung 7.10: Modell der Korrelationsdaten

Klasse *CorrelationEntry* beschreibt Korrelationswerte. Jedem *CorrelationEntry* ist neben dem eigentlichen Korrelationswert (*CorrelationScore*) eine *LocalConnection* und eine *NamedConnection* zugeordnet. Diese beschreiben jeweils, zu welcher Datenanforderung und welchem externen Sensor die Klasse gehört. Zusätzlich zu den Konzepten ist auch ein beispielhafter Eintrag dargestellt: *West* und *BEastIn* wird ein Korrelationswert von 0.87 zugewiesen. *BEastIn* taucht an dieser Stelle als Individuum und nicht als Klasse auf. *BEastIn* muss hier als Individuum verwendet werden, damit direkt ein Korrelationswert zugeordnet werden kann. In den anderen Zusammenhängen muss *BEastIn* eine Klasse sein, damit Instanzen von *BEastIn* erzeugt werden können, um konkrete Messwerte abzubilden. Eigentlich ist das in OWL (wie auch in UML) nicht zulässig - Elemente der ABox können nicht Teile der TBox sein. An dieser Stelle wird deshalb eine als „Punning“ bezeichnete Technik verwendet: der Klasse und dem Individuum werden die selbe URI zugewiesen und der Reasoner unterscheidet je nach syntaktischem Kontext, ob von einer Klasse oder von einem Individuum gesprochen wird [Mot07]. Diese Korrelationsdaten existieren für alle Sensoren. Die Daten werden genutzt, um bei einer Änderung das Alignment zwischen der lokalen und globalen Ontologie anzupassen.

Reaktion auf Veränderungen

Da das Wegfallen und Hinzukommen von Datenquellen eine Anpassung von Alignment-Statements erfordert, kann der in Abschnitt 5.1 beschriebene Ansatz zum Einbeziehen von Inferenzen verwendet werden. Bei einer Veränderung in der globalen oder lokalen Ontologie werden die lokale und globale Ontologie sowie das Alignment zwischen den Ontologien als ein Graph betrachtet. G ist der Graph vor der Veränderung und G' ist der Graph nach der Veränderung, und C ist die Menge der Veränderten Knoten in G. Wie in Definition 5.1.5 beschrieben, werden die Menge der hinzugekommenen und entfernten Inferenzen Δ_{del} und Δ_{add} zu den Veränderungen C berechnet.

Um festzustellen, ob ein Änderungsbedarf besteht, wird passend zu den in Abschnitt 5.1.3 definierten Funktionen überprüft, ob ein Knoten in den hinzugekommenen oder entfernten

Inferenzen eine Unterklasse der Klasse *External* ist. Ist dies der Fall, muss möglicherweise eine Anpassung vorgenommen werden. Sowohl das Entfernen als auch das Hinzufügen von Klassen wird gleich behandelt, damit auch beim Hinzufügen von neuen Sensoren überprüft wird, ob gut passende Datenquellen existieren. Es werden also zwei identische Funktionen wie folgt implementiert:

$$\gamma_{add}(c) = \gamma_{del}(c) := \begin{cases} \boxed{a} & \text{wenn ein } e \in E \text{ existiert mit } s(e) = c, \text{ die Klasse External wird durch den Knoten } t(e) \text{ repräsentiert und } \Lambda(e) \text{ repräsentiert die Unterklassenbeziehung} \\ \boxed{n} & \text{sonst.} \end{cases}$$

Auf alle Knoten c, die mit $\boxed{a}$ markiert wurden, werden dann zwei weitere Funktionen angewandt: (1) Die Funktion zur Berechnung der Graph-Entfernungen α^- berechnet das Alignment-Statement, das c mit einem Informationsbedarf verbindet. Diese Statements werden entfernt, um Platz für einen Ersatz zu schaffen.

(2) Die Funktion zur Berechnung der Graph-Ergänzungen α^+ wendet das in Quellcode 7.1 dargestellte SPARQL-Query an, um Kandidaten für neue Tripel zu erzeugen. In Zeile 3 wird `c.URI` durch die URI des Graph-Knotens ersetzt, mit dem c durch ein Alignment-Statement verbunden war. Das SPARQL-Query gibt ein Tupel aus Klasse und Korrelationswert zurück, wenn diese durch einen passenden *Correlation Entry* an c gebunden sind. Diese Tupel werden nach Korrelationswert geordnet.

```
1 SELECT ?dataClass ?score WHERE {
2         ?corrItem :namedItem ?dataClass;
3                 :localItem <"c.URI">;
4                 :correlationScore ?score
5 } ORDER BY DESC(?score)
```

Listing 7.1: SPARQL Query zur Berechnung von Kandidaten für neue Tripel

Das Ergebnis des SPARQL-Queries wird dann verwendet, um das neue Alignment-Tripel zu berechnen. Der erste Eintrag e im ersten Tupel der Ergebnisliste, der sich nicht auf einen entfernten Knoten bezieht, wird dann weiterverwendet, um ein neues Alignment-Tripel zu erzeugen. Das zu ergänzende Alignment-Tripel ist `e rdfs:subClassOf c.URI`. Wenn die Ergebnisliste leer ist, wird kein neues Tripel hinzugefügt. Mit diesen Funktionen lässt sich der Ansatz zur Einbeziehung von Inferenzen vollständig anwenden, er kann genau so angewendet werden wie der in Abschnitt 5.1 beschriebene Algorithmus 1.

Dieser Prozess würde, wenn in dem in Abbildung 7.9 dargestellten Beispiel *BEastIn* entfernt

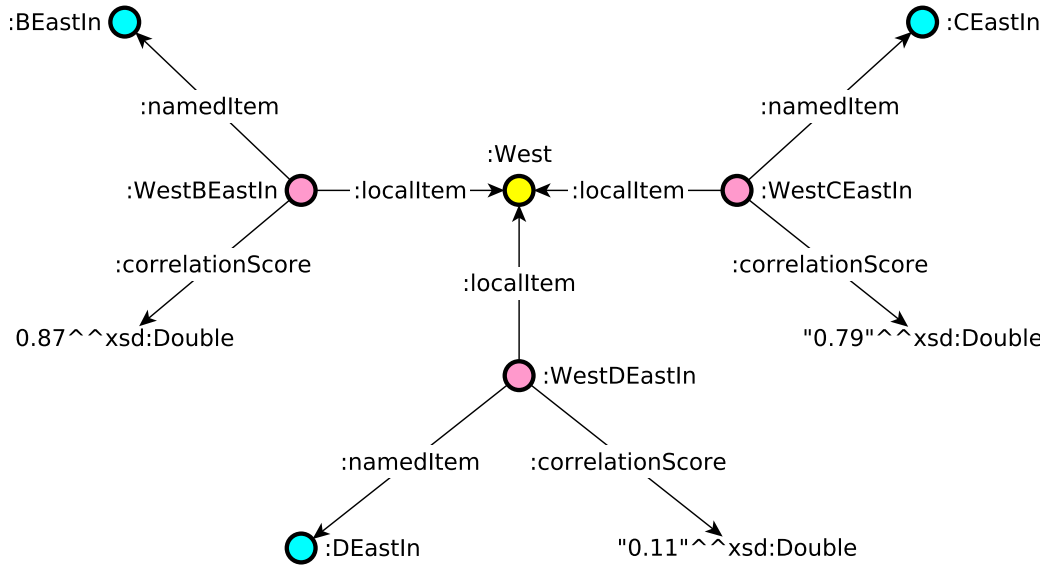

Abbildung 7.11: Beispielhafte Korrelationsdaten

Tabelle 7.1: Ergebnisse des SPARQL-Queries für die in Abbildung 7.11 dargestellte Ontologie

?dataClass	?score
:BEastIn	0.87
:CEastIn	0.79
:DEastIn	0.11

wurde, wie folgt ablaufen: Zunächst wird die Funktion γ_{add} angewandt. Die Funktion würde feststellen, dass *BEastIn* eine Unterklasse von *External* ist. Also ist eine Reaktion notwendig und Funktion (1) gibt $\boxed{a}$ zurück. Also werden die Funktionen α^+ und α^- auf *BEastIn* angewandt. Die ABox einer beispielhaften lokalen Korrelationsontologie ist in Abbildung 7.11 dargestellt. Mit dieser Korrelationsontologie wird das in Listing 7.1 dargestellte SPARQL-Query als Ergebnis Tabelle 7.1 erzeugen. Da `:BEastIn` herausgefiltert wird, wird `:CEastIn` als passendster Kandidat für das neue Alignment-Tripel klassifiziert. Das Ergebnis der Funktion und damit das neue Tripel ist also `:CEastIn rdfs:subClassOf :West`.

7.2.4 Proof of Concept

Dass der dargestellte Ansatz grundsätzlich funktionieren kann, wurde bereits durch die abstrakte Formulierung des Ansatzes und die Definition der Funktionen, die die passenden Änderungen berechnen, gezeigt. Um zu untersuchen, welche Eigenschaften der Ansatz bei der Umsetzung aufweist, wurde außerdem ein Prototyp entwickelt. Die Ontologien werden mithilfe der Ontolo-

giesprache OWL 2 [Hit+09] beschrieben. Die Verwaltung und Bearbeitung der Ontologien wird mit dem Apache Jena-Framework umgesetzt [Apa20]. Als Programmiersprache wurde Scala genutzt, da sich so die funktionale Berechnung der Anpassungsfunktionen mit der Verwendung von populären Ontologie-APIs verbinden ließ. Der Prototyp liest zwei Versionen eines Graphen ein. Der Graph besteht also aus der lokalen Ontologie, der globalen Ontologie und dem Alignment. Aus den Eingabedaten werden dann die notwendigen Veränderungen des Alignments berechnet. Die domänenspezifischen Aspekte der Implementierung wurden über Hooks in das generische Framework eingegliedert. So musste nur wenig Code geschrieben werden, der spezifisch für den Anwendungsfall ist.

Der Prototyp wurde anhand von konstruierten Beispielen getestet. Sowohl für das Entfernen als auch das Hinzufügen von Ampelsteuerungen und Sensoren wurden Testfälle mit erwarteten Ergebnissen konstruiert. Das System lieferte in allen Fällen die erwarteten Ergebnisse. Es ist also möglich, mit der hier vorgestellten Methode ein korrekt funktionierendes System zu erzeugen.

7.2.5 Ergebnisse

Der domänenspezifische Teil der Anwendung besteht aus weniger als 100 Zeilen Source-Code, der Rest wird in zum größten Teil domänenunabhängigen OWL-Ontologien und SPARQL-Anfragen abgebildet oder durch das zugrunde liegende Framework bereitgestellt. Ein großer Teil der Anwendung, die Ontologien, lässt sich also generieren oder mit erprobten Standard-Tools erstellen. So können Domänenexperten einen großen Teil der Lösung erzeugen, ohne über Programmierkenntnisse zu verfügen. Außerdem gibt es zu Tools zum Erstellen von Ontologien wie Protégé umfangreiche, öffentlich verfügbare Dokumentation und eine hilfsbereite und aktive Community. Also können Domänenexperten beim Lernen der Modellierungssprache mit Ressourcen aus der Community unterstützt werden. Die Struktur dieses Ansatzes erlaubt es den Entwicklern, sich auf die Invarianten und Regeln konzentrieren, die für den Anwendungsfall tatsächlich wichtig sind.

Der dargestellte Anwendungsfall ist nicht die einzige Möglichkeit, dieses Verfahren anzuwenden. Es wären auch Übertragungen auf andere Arten von Verkehrssteuerungen, zum Beispiel im Luftverkehr oder im Schienenverkehr denkbar. Auch bei verwandten Problemen könnte das Verfahren angewandt werden, zum Beispiel beim Paket-Routing oder ähnlichen Lastverteilungsproblemen, bei denen Verteilungsentscheidungen an einem Knotenpunkt die Situation an anderen Knotenpunkten beeinflussen.

Die erfolgreiche Implementierung zeigt auch, dass sich trotz der strengen Einschränkungen aus Abschnitt 5.1.3 ein Anwendungsfall mit realistischen Aufgaben mit geringem Programmieraufwand umsetzen lässt. Dass diese Einschränkungen deutlich strenger gewählt wurden als dies

für den Einsatz des Ansatzes tatsächlich notwendig ist spricht dafür, dass der regelbasierte Ansatz auf weitere Anwendungsfälle in weiteren Domänen angewandt werden kann.

Dieser Ansatz ist allerdings nur einsetzbar, wenn die Modellierung so gestaltet werden kann, dass die entsprechenden Voraussetzungen erfüllt sind: Das Ontologiemodell und alle Regeln wurden genau an den Anwendungsfall angepasst. Wenn Informationen im Modell fehlen, kann dieser Ansatz nicht angewandt werden. Wären zum Beispiel im dargestellten Anwendungsfall keine Korrelationsscores vorhanden, müsste auf eine andere Art bestimmt werden, wie Ersetzungen vorgenommen werden können. Es ist außerdem eine wichtige Voraussetzung, dass Regeln für die Veränderungen des Alignments gefunden werden können: Der Ansatz kann auch nicht eingesetzt werden, wenn es prinzipiell nicht möglich ist mit den vorhandenen Informationen Regeln für die nötigen Anpassungen an Veränderungen zu finden, oder wenn der Aufwand um diese Regeln zu finden deutlich zu hoch wäre. In solch einem Fall können Machine-Learning-Ansätze angewandt werden, wie im nun folgenden Abschnitt beschrieben wird.

7.3 Ansätze, die maschinelles Lernen einsetzen

In diesem Abschnitt wird die Evaluation von drei Ansätzen behandelt: die Evaluation des Ansatzes, der Embeddings verwendet (siehe Abschnitt 5.2), die Evaluation des Graph-Convolution-basierten Ansatzes (siehe Abschnitt 5.4) und die Evaluation des Ansatzes, der ein Veränderungsmodell mit maschinellem Lernen kombiniert (siehe Abschnitt 5.3). Diese Ansätze werden gemeinsam untersucht, da sie sich gut auf einem Datensatz gemeinsam betrachten und vergleichen lassen. Der Datensatz, der dazu verwendet wird, stammt aus dem biomedizinischen Bereich. Außerdem verfolgen die Ansätze die gleiche Aufgabenstellung – die Vorhersage von von Veränderungen betroffenen Alignment-Statements. Die hier vorgestellten Ergebnisse sind teilweise bereits als Beiträge zu Workshops veröffentlicht worden [JI18; JI19]

Folgende Fragestellungen sollen mit der Evaluation betrachtet werden:

1. Lassen sich mit diesem Verfahren Ergebnisse bezüglich üblicher Klassifikationsmetriken erzielen, die besser sind als nicht-lernende Methoden wie zufälliges Auswählen einer Klasse oder das statische Vorhersagen der in der Mehrheit vorhandenen Klasse?

2. Welchen Einfluss haben die Wahl des Klassifikationsverfahrens und Wahl des Embeddingverfahrens auf die Ergebnisse bezogen auf verbreitete Klassifikationsmetriken?

3. Wie ist die Leistung der Embedding-basierten Verfahren im Vergleich zu Verfahren, die auf Graph Convolution basieren?

4. Welchen Effekt hat die Einbeziehung von Veränderungsmodellen?

7.3.1 Datensatz

Es gibt zwar viele Anwendungsmöglichkeiten für Ontologie-Alignments, allerdings gestaltet sich die Suche nach öffentlich verfügbaren Daten aufgrund der in Abschnitt 7.1 dargestellten Anforderungen als nicht einfach: In vielen Fällen sind zu Ontologien jeweils nur die neusten Versionen veröffentlicht, oder es gibt kein Alignment. Und wenn es ein Alignment gibt, liegt es häufig nur in einer Version vor. Eines der wenigen öffentlich verfügbaren Beispiele ist ein aus dem Unified Medical Language System (UMLS) extrahierter Datensatz, der durch die Autoren von [Gro+13] veröffentlicht wurde[1] Erstmalig wurde der Datensatz in [Jr+10] beschrieben. Der UMLS ist ein sogenannter Metathesaurus, der mehr als 60 verschiedene Begriffssysteme aus der biomedizinischen Forschung zusammenfasst [Bod04]. Ein Metathesaurus ist eine Menge von semantischen Verbindungen zwischen verschiedenen Begriffssystemen. Der UMLS Metathesaurus besteht seit 1986 und soll die Kommunikation zwischen biomedizinischen Forschern erleichtern und ist dem entsprechend auch öffentlich zugänglich. Die Informationen in UMLS werden händisch erstellt und zweimal jährlich veröffentlicht [Bod04]. Reasoning auf dem UMLS gestaltet sich allerdings als nicht besonders einfach. Der Datensatz ist sehr groß und enthält einige Fehler. Dementsprechend ist es sehr schwierig, aus den Inferenzen in diesem Datensatz manuell allgemeingültige Regeln abzuleiten. Aus diesem Grund lassen sich auf diesem Datensatz nur die Verfahren aus dieser Arbeit, die maschinelles Lernen anwenden, einsetzen.

Der UMLS wird als Grundlage für die Evaluation verwendet. Aus UMLS wird dabei hier nur ein Ausschnitt betrachtet: es liegen jährliche Versionen aus den Jahren 2009-2012 für drei Teilontologien und Alignments zwischen diesen Ontologien vor. Dieser Ausschnitt wird gewählt, weil er in anderen Arbeiten auch verwendet wurde [Gro+13; Jr+10] und weil er in dieser Form veröffentlicht wurde. Die drei Teilontologien sind *Systematized Nomenclature of Medical Descriptions (SNOMED)*, *Foundational Model of Anatomy (FMA)* und der *National Cancer Institure Thesaurus (NCI-T)*. SNOMED ist eine Sammlung von medizinischen Termen, wobei der Fokus auf der systematischen Erfassung von Patientendaten liegt [Ben12]. FMA beschreibt die menschliche Anatomie in einer Ontologie [RMJ03]. Der NCI Thesaurus enthält Informationen und Konzepte zu Krebs aus einer Behandlungsperspektive und auf molekularer Ebene [Sio+07]. Alle Ontologien haben also einen Bereich, in dem sie sich überschneiden oder zumindest für die jeweils anderen Ontologien relevant sind. Genauso gibt es aber auch Ausschnitte, die mit den anderen Ontologien nicht viel zu tun haben. Beispielsweise enthalten sowohl SNOMED-CT als auch NCI-T Konzepte, die mit der Behandlung von Krebs zu tun haben. Gleichzeitig enthält SNOMED aber auch Terme, die nichts mit der Behandlung von Krebs zu tun haben und NCI-T

[1] https://dbs.uni-leipzig.de/de/research/projects/evolution_of_ontologies_and_mappings/ontology_mapping_adaption

Konzepte, die nicht für die Behandlung von Patienten relevant sind.

Die Daten aus diesem Datensatz werden als eine Art Silver-Standard verwendet: die Daten können zwar Fehler enthalten, es wird aber davon ausgegangen, dass sie im wesentlichen korrekt sind und zum Training und zur Evaluation von Algorithmen verwendet werden können, da keine besseren Daten zur Verfügung stehen. Auch in anderen Arbeiten (zum Beispiel [Gro+13; Jr+10] wird so vorgegangen. Da nur die Rohdaten (also die Ontologie- und Alignmentversionen ohne eine Klassifikation von Veränderungen) zur Verfügung stehen, musste der Datensatz noch aufbereitet werden. So müssen die Veränderungen noch identifiziert und in zwei Klassen aufgeteilt werden. Als Trainingsdatensatz werden die Veränderungen von 2009-2011 verwendet und zur Evaluation die Veränderungen von 2011-2012. Es stehen also zwei Versionssprünge zum Trainieren zur Verfügung und einer zum Evaluieren. Veränderungen in den Daten werden mithilfe von OWL-Diff [Kre+11] erkannt, einem Plugin für die verbreitete Ontologie-Suite Protégé [Noy+01].

Veränderungen an den Ontologien werden entsprechend der Definition 5.2.1 in Veränderungen, die eine Veränderung am Alignment nach sich ziehen (markiert mit $\boxed{c}$) und Veränderungen die keinen Einfluss auf das Alignment haben (markiert mit $\boxed{n}$) klassifiziert. Es werden nur Veränderungen betrachtet, die über maximal zwei Kanten mit einem an einem Alignment-Statement beteiligten Knoten erreichbar sind, da alle veränderten Knoten, die mehr als zwei Kanten vom Alignment entfernt immer in der Klasse $\boxed{n}$ sind, da in ihrer jeweiligen zwei Schritte entfernten Nachbarschaft keine Alignment-Statements vorkommen.

Würden alle veränderten Knoten betrachtet, müssten sehr viele Veränderungen betrachtet werden, die größtenteils nichts mit dem Alignment zu tun haben. Dieses große Ungleichgewicht würde einen starken Einfluss auf die Klassifikation haben. Da ein großer Teil der Veränderungen nicht in direkter Umgebung des Alignments stattfindet und so auch keinen Einfluss hat, wäre der Datensatz extrem unausgeglichen. Außerdem wäre die Vorhersage von anzupassenden Alignment-Knoten sehr einfach, da die Entfernung zu einem Alignment-Knoten ein sehr starkes Vorhersagekriterium wäre.

Zur Erstellung des Datensatzes wird die Erweiterung eines Protégé-Plugins zur Erkennung von Ontologie-Veränderungen verwendet. Das in Abschnitt 6.1.1 beschriebene Plugin wird verwendet, um die zwischen zwei Ontologieversionen veränderten Knoten zu erkennen und in einer Datei abzuspeichern. Für die in dieser Datei gespeicherten Veränderungen wird mit dem in Abschnitt 6.1.2 vorgestellten Framework berechnet, welche dieser Veränderungen über zwei Kanten ein Alignment-Statement erreichen. Für diese Veränderungen werden mit dem gleichen Framework die Umgebungen der ausgewählten Veränderungen verglichen. Es wird untersucht, ob sich in dieser Umgebung Alignment-Statements verändert haben.

Es werden jeweils Paare von Ontologien mit den entsprechenden Aligenments betrachtet. In Tabelle 7.2 ist eine Quantifizierung der Kategorisierung abgebildet, wenn nur Knoten in der Nähe des Alignments betrachtet werden. Dabei bezeichnet entsprechend Definition 4.3.1 V die Menge der Knoten in dem Graph aus beiden Ontologien und dem Alignment und E die Kanten. C_{change} bezeichnet die Menge aller Veränderungen, die eine Veränderung am Alignment nach sich ziehen, also $C_{change} := \{v|\lambda(v) = \boxed{c}\}$. Dementsprechend bezeichnet die Menge $C_{nochange}$ die Veränderungen, die keinen Einfluss auf das Alignment haben, also $C_{nochange} := \{v|\lambda(v) = \boxed{n}\}$. Es ist deutlich zu erkennen, dass die Verhältnisse von C_{change} und $C_{nochange}$ teilweise sehr unterschiedlich sind. Auch zwischen Trainings- und Testmenge unterscheiden sich die Größen der Mengen teilweise sehr. Die Größe der Ontologien selbst unterscheidet sich zwischen den Versionen nur geringfügig.

Tabelle 7.2: Datensatz

	Version	$\#V$	$\#E$	$\#C_{change}$	$\#C_{nochange}$
FMA-NCI	2009-2011	2M	7M	725	984
	2011-2012			352	421
SNOMED-FMA	2009-2011	4M	15M	1526	13435
	2011-2012			177	6925

Die dargestellten Daten werden verwendet, um die Evaluation durchzuführen, auch wenn die Daten nicht immer ausgeglichen sind, da dies dem realistischen Anwendungsfall entspricht. Auf diesen Umstand muss bei der Betrachtung der Metriken Rücksicht genommen werden.

Es erfolgt an dieser Stelle kein quantitativer Vergleich mit anderen Arbeiten, da ein direkter Vergleich nicht möglich ist. Auch die Ergebnisse der Arbeiten, die auf dem gleichen Datensatz arbeiten, sind nicht direkt vergleichbar, da diese in Abschnitt 3.1 beschriebenen Arbeiten ein grundsätzlich anderes Ziel verfolgen (die Vorhersage eines vollständigen Alignments). Diese Ansätze werden anhand der Übereinstimmung des vorhergesagten Alignments mit dem tatsächlichen Alignment evaluiert. Da ein sehr großer Teil der Alignments von Veränderungen allerdings gar nicht betroffen sind, ist eine aussagekräftige Evaluation für diese Zielsetzung mit den hier gewählten Ansätzen nur schwer möglich. Insbesondere ist das Ziel der in dieser Arbeit vorgestellten Ansätze die Vorhersage von betroffenen und nicht betroffenen Alignment-Statements, während die Ansätze aus der Literatur die nicht betroffenen Alignments als gegeben ansehen.

7.3.2 Bewertung der Ergebnisse

Die Daten werden wie im vorherigen Abschnitt beschrieben in Test- und Trainingsdaten aufgeteilt. Dann werden die Ansätze zur Klassifikation angewandt, wie in den Abschnitten 5.2 und 5.4 beschrieben. Um die Klassifikation zu beurteilen, werden mehrere verbreitete Klassifikationsmetriken verwendet. Die verwendeten Metriken sind *accuracy, precision, recall, f1-measure, die Fläche unter der ROC-Kurve* und *average precision*. Für eine Definition dieser Metriken siehe Abschnitt 2.3.3.

7.3.3 Umsetzung - Embeddings

Für den Embedding-basierten Ansatz werden die Embeddings auf dem Wissensgraph trainiert. Für die Embedding-Verfahren aus dem Knowledge-Graph-Completion-Bereich (TransE, TransH, DistMult, ComplEX) wurde eine Implementierung mithilfe des OpenKE -Framewokrs [Han+18] umgesetzt. Zur Erzeugung der RDF2Vec-Embeddings wird die Implementierung der Autoren von [RP16] erweitert [2]. Die Hyperparameter dieser Verfahren wurden entsprechend der Empfehlungen in der Dokumentation der Verfahren gewählt.

Die Klassifikation wurde mit dem verbreiteten Python-Framework scikit-learn [Ped+11] implementiert. Die Umsetzung der Klassifkatoren und Embeddings sind in Abschnitt 6.2 vorgestellt. Es wurden Klassifikationsverfahren aus verschiedenen Bereichen angewandt, alle Verfahren sind in Tabelle 5.2 aufgeführt. Um die beste Kombination von Embedding- und Klassifikationsverfahren zu finden, wurden alle Kombinationen evaluiert. Dies führte zu mehr als 40 Kombinationen.

Jedes Klassifikationsverfahren wird auf beiden Datensätzen, FMA-NCI und SNOMED-FMA angewandt. Eine Ausnahme stellt RDF2Vec dar: dieses Verfahren konnte nicht auf SNOMED-FMA angewandt werden. Die Schwierigkeit stellten hier die Random Walks dar. Bei der Anwendung von RDF2Vec werden zuerst Dateien mit Random Walks erzeugt. Diese Random Walks werden dann als Eingabe für Word2Vec verwendet. Der SNOMED-FMA Graph etnhält insgesamt vier Millionen Entitäten und 15 Millionen Tripel. Bei 200 Random Walks je Entität wurden die Random Walks so groß, dass sie mit der zur Verfügung stehende Hardware nicht mehr zu verarbeiten waren. Weniger Random Walks je Entität zu generieren würde aber zu deutlich schlechteren Ergebnissen führen, da für viele Entitäten so keine sinnvolle Repräsentation mehr gefunden wird. Für FMA-NCI ist das noch möglich, aber auch hier werden die Random Walks schon sehr groß.

[2] http://data.dws.informatik.uni-mannheim.de/rdf2vec/code/

7.3.4 Umsetzung - Graph Convolutional Networks

Für den Graph-Convoloution-basierten Ansatz wurde die RGCN-Implementierung verwendet, die von den Autoren von [Sch+18] auf GitHub veröffentlicht wurde[3]. Als Hyperparameter wurden 5 hidden Layer, ein l2 penalty von 0,005, eine dropout rate von 0.005, eine Lernrate von 0.01 und 50 Trainingsepochen ausgewählt. Es wurden nur 5 hidden layer verwendet, da das Neuronale Netz so gerade noch in den Arbeitsspeicher passt. Die anderen Parameter wurden basierend auf einer Gitter-Suche über den Parameterraum gefunden.

7.3.5 Ergebnisse für Embeddings und Graph Convolution

Ergebnisse der Evaluation für den Datensatz FMA-NCI sind in Tabelle 7.3 dargestellt. Die unterstrichenen Einträge repräsentieren jeweils den besten Wert für eine Metrik. Es werden aus Platzgründen nicht alle Kombinationen von Embedding-und Klassifikations-Verfahren dargestellt, Stattdessen werden für jedes Embedding-Verfahren die zwei besten Klassifikationsmethoden in Bezug auf f1-Measure angegeben. Da Graph Embeddings nicht deterministisch sind, wurden die Berechnung der Embeddings wiederholt. Da sich keine signifikante Veränderung der Klassifikationsergebnisse gezeigt hat, werden die Ergebnisse mit der ersten Version der Embeddings dargestellt.

Die Kombination aus dem Embedding-Verfahren RDF2Vec mit logistischer Regression (LR) als Klassifikationsverfahren erreichte für alle Metriken außer Precision und die Fläche unter der ROC-Kurve den besten Wert. Mindestens eine der anderen Kombinationen aus Embedding-Verfahren und Klassifikationsverfahren erreicht bezogen auf eine Metrik ähnliche Werte, keine jedoch auf alle. Der RGCN-Ansatz erreicht beim ROC-Wert und Accuracy bessere Werte als RDF2Vec, bei allen anderen Metriken erreicht RDF2Vec bessere Ergebnisse. Die accuraccy und precission von fast allen Verfahren ist größer als 0.54. 0.54 (der Anteil der größeren Klasse $C_{nochange}$ im Datensatz) wäre der Wert, der beim Annehmen der Merhheitsklasse oder dem zufälligen Zuordnen von Beispielen zu Klassen erreicht werden könnte. Dementsprechend funktionieren die vorgestellten Verfahren besser als solche statischen Klassenzuweisungen.

Ergebnisse für den Datensatz SNOMED-FMA werden in Tabelle 7.4 dargestellt. Die Metriken sind in allen Fällen deutlich niedriger als auf dem FMA-NCI-Datensatz. Dies ist vor allem durch die Verteilung der Daten im Testdatensatz begründet: nur 2,6% der Beispiele sind in der Klasse C_{change}. Auch im Trainingsdatensatz ist die Klasse C_{change} mit 10,2% unterrepräsentiert. Dementsprechend ist es deutlich schwerer, diese Klasse vorherzusagen, als dies in einem ausgeglichenen Datensatz der Fall ist. Außerdem ist, wie bereits erwähnt, keine Kombination mit

[3] https://github.com/tkipf/relational-gcn

Tabelle 7.3: FMA-NCI Results

Embedding	Klasisfikation	f1	acc	prec	rec	roc_auc	avg. prec
TransE	KNN	0.587	0.659	0.642	0.541	0.648	0.553
	RF	0.551	0.659	0.672	0.467	0.641	0.553
	SVM(RBF)	0.265	0.602	0.771	0.160	0.561	0.500
	MLP	0.550	0.683	0.756	0.432	0.659	0.582
TransH	MLP	0.555	0.655	0.659	0.479	0.638	0.549
	RF	0.567	0.657	0.655	0.500	0.643	0.552
DistMult	RF	0.516	0.639	0.647	0.429	0.619	0.534
	MLP	0.560	0.656	0.657	0.488	0.640	0.551
Complex	MLP	0.572	0.611	0.565	0.580	0.608	0.516
	LR	0.398	0.542	0.485	0.337	0.523	0.461
RDF2Vec	NB	0.657	0.548	0.701	0.619	0.501	0.701
	LR	0.735	0.647	0.774	0.700	0.611	0.752
RGCN		0.6247	0.778	0.706	0.561	0.723	0.540

RDF2Vec in dieser Evaluation enthalten.

Auf SNOMED-FMA gibt es keine Kombination aus Embedding- und Klassifikationsverfahren, die besser abschneidet als das RGCN-basierte Verfahren. RGCN erreicht in fast allen Metriken signifikant bessere Ergebnisse als die anderen Klassifikatoren. Die einzige Ausnahme hier ist Accuracy, hier ist TransE kombiniert mit logistischer Regresssion fast genau so gut. Allerdings ist eine gute Accuracy auf diesem Datensatz auch sehr einfach zu erreichen – Ein Verfahren, dass alle Beispiele in der Kategorie $C_{nochange}$ einsortiert, würde eine Accuracy von 97,4% erreichen, genau wie TransE mit logistischer Regression. Betrachtet man den Recall-Wert für diesen Fall kann man feststellen, dass die Kombination aus TransE mit logistischer Regression so arbeitet. Mit Bezug zum Vergleich zur statischen Zuweisung von Klassen kann festgestellt werden, dass RGCNs sowohl die Accuracy als auch die Precision von solchen Verfahren überbietet. Dieses Verfahren erzielt also bessere Ergebnisse als statische Zuweisungen.

Insgesamt scheinen die Ergebnisse, die mit RGCN erzeugt werden, in vielen Bereichen die besten zu sein und in anderen Bereichen sehr nah an den besten Ergebnissen zu liegen. Ins-

Tabelle 7.4: SNOMED-FMA Results

Embedding	Klassifikation	f1	acc	prec	rec	roc_auc	avg. prec
TransE	LR	0.070	0.974	0.318	0.040	0.519	0.037
	NB	0.155	0.857	0.091	0.525	0.695	0.060
	KNN	0.155	0.965	0.192	0.130	0.558	0.047
	RF	0.189	0.944	0.148	0.260	0.611	0.057
TransH	MLP	0.248	0.966	0.276	0.226	0.605	0.082
	RF	0.191	0.944	0.149	0.266	0.613	0.058
Distmult	MLP	0.193	0.943	0.150	0.271	0.616	0.059
	RF	0.221	0.943	0.168	0.322	0.641	0.071
Complex	RF	0.243	0.944	0.183	0.362	0.660	0.082
	MLP	0.221	0.937	0.160	0.356	0.654	0.073
RGCN		0.542	0.978	0.508	0.581	0.784	0.305

besondere ROC-AUC und Accuracy sind in beiden betrachteten Anwendungsfällen besser als die anderen betrachteten Ansätze. RDF2Vec scheint ähnlich gute Ergebnisse produzieren zu können, konnte allerdings im zweiten Anwendungsfall nicht eingesetzt werden, da zu viel Speicherplatz für Random Walks benötigt werden würde.

7.3.6 Ergebnisse bei Over- und Undersampling

Um den hier verwendeten Ansatz an den nicht ausgeglichenen Datenansatz anzupassen, wurden noch zwei weitere Experimente auf dem SNOMED-FMA-Datensatz durchgeführt:

1. *Oversampling der unterrepräsentierten Klasse:* Beispiele aus der unterrepräsentierten Klasse (C_{change}) werden mehrfach beim Training betrachtet, so dass beide Klassen beim Training gleich häufig vorkommen.

2. *Undersampling der überrepräsentierten Klasse:* Beispiele aus der überrepräsentierten Klasse ($C_{nochange}$) werden beim Training ignoriert, so dass beide Klassen beim Training gleich häufig vorkommen.

In Tabelle 7.5 sind Ergebnisse dieser Experimente dargestellt. Es werden nur die Ergebnisse mit dem höchsten f1-Measure für jedes Embedding-Verfahren und das RGCN aus beiden Experimenten abgebildet. Die besten Ergebnisse für die Embedding-Basierten Verfahren wurden mit Oversampling erzeugt, die besten Ergebnisse für RGCN wurden mit undersampling erzeugt. Die f1-Measure Werte sind insgesamt ähnlich zu den Ergebnissen ohne over-/unersampling. Erwartungsgemäß sind die Recall-Werte deutlich erhöht, während die Precision-Werte deutlich niedriger sind als mit der herkömmlichen Sampling-Strategie.

Tabelle 7.5: SNOMED-FMA Over/Undersampling

Embedding	Klassifikation	f1	acc	prec	rec	roc_auc	avg. prec
TransE	RF	0.222	0.896	0.137	0.598	0.751	0.091
TransH	CART	0.224	0.896	0.138	0.605	0.754	0.093
Distmult	MLP	0.223	0.895	0.136	0.605	0.753	0.092
Complex	MLP	0.222	0.895	0.136	0.599	0.751	0.091
RGCN		0.461	0.965	0.355	0.656	0.814	0.241

7.3.7 Umsetzung - Verbindung von maschinellem Lernen und Veränderungsmodellen

Zur Umsetzung der Verbindung von maschinellem Lernen und Veränderungsmodell wird neben den bereits diskutierten Embeddings noch ein Veränderungsmodell benötigt. Dazu wird die in Abschnitt 6.1.1 vorgestellte Erweiterung des owl-diff-Plugins verwendet. Die Erweiterung greift intern auf das Veränderungsmodell von owl-diff zurück und kann zu einer Veränderung Tags speichern, die festhalten, wie ein Knoten verändert wurde. Das Plugin unterstützt 5 Arten von Veränderungen: Löschen von Statements zu Klassen (*Deleted*), Hinzufügen von Statements zu Klassen (*Added*), Ändern der Oberklasse (*Superclass*), Veränderung einer Annotation (*Annotation*) und Umbenennung der Klasse (*Renamed*). Da hier nur eine T-Box-Ontologie betrachtet wird, beziehen sich alle Veränderungen auf Klassen. In Abbildung 7.12 sind die relativen Häufigkeiten der einzelnen Veränderungsklassen dargestellt. Die häufigsten Veränderungsklassen sind *Deleted* und *Added*. *Superclass* und *Annotation* kommen seltener vor. Die Klasse *Renamed* kommt in dem Datensatz nicht vor.

Die gespeicherten Tags werden, wie in Abschnitt 5.3.2 beschrieben, vewendet, um die Funktion δ abzubilden. Die bereits in Abschnitt 7.3.3 vorgestellten Embeddings werden verwendet um die Funktion τ zu repräsentieren. Auch die Klassifiktaionsalgorithmen, die verwendet wer-

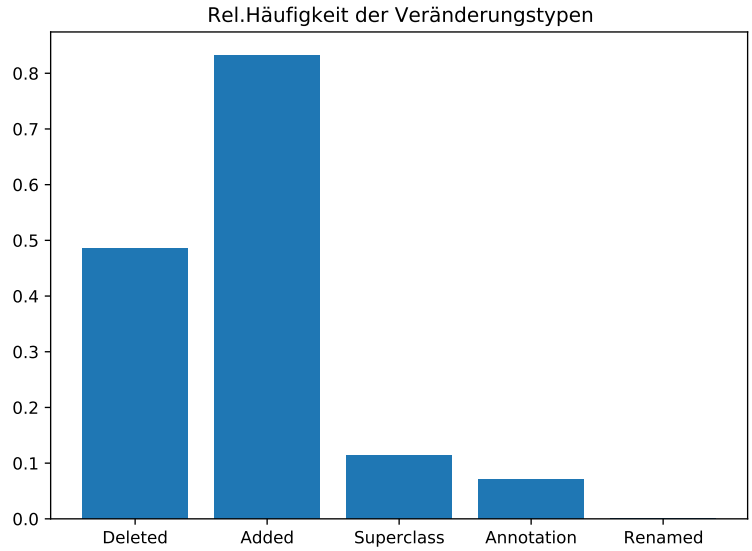

Abbildung 7.12: Relative Häufigkeit der Veränderungen FMA-NCI

den, sind die gleichen. Die damit erzeugten Trainingsdaten unterscheiden sich allerdings: Die Trainingsdaten beziehen sich nur auf den Versionssprung von 2009 nach 2010 und die Testdaten auf die Versionsveränderung von 2010 nach 2011. Dieser kleinere Datensatz wurde gewählt, um in einer Vorevaluation festzustellen, ob der gewählte Ansatz Vorteile bieten kann. Die Evaluation kann auf dem kleinen Datensatz schneller durchgeführt werden als auf dem Datensatz, der in Abschnitt 7.3.1 beschrieben wird.

7.3.8 Ergebnisse für die Kombination von Veränderungsmodell und Embeddings

Ein Ausschnitt der Ergebnisse für die Kombination von Embeddings und Veränderungsmodell ist in Tabelle 7.6 dargestellt. Es wurden insgesamt 64 Kombinationen aus Embedding-Verfahren und Klassifikationsverahren jeweils mit und ohne Veränderungsmodell untersucht. In der Tabelle sind die Metriken der trainierten Modelle mit und ohne Veränderungsmodell dargestellt. Die Ergebnisse für die die Kombination aus Veränderungsmodell und Embeddings sind mit $+VM$, die ohne Veränderungsmodell mit $-VM$ bezeichnet. Für jede Metrik sind die zwei besten Ergebnisse für eine Konfiguration aus Embedding-Verfahren und Klassifikationsverfahren mit Veränderungsmodell dargestellt. Um die Unterschiede zwischen der ausschließlichen Verwendung von Embeddings und der Kombination von Embeddings und Veränderungsmodell hervorzuheben, sind die entsprechenden Konfigurationen von Embedding und Klassifikationsverfahren zusätzlich dargestellt. Die besten Ergebnisse für jede Metrik sind jeweils unterstrichen.

Bei näherer Betrachtung dieser Ergebnisse fällt auf, dass sich die Ergebnisse mit und oh-

Tabelle 7.6: FMA-NCI Kombination von Embeddings und Veränderungsmodell

Features	Klassifikation	f1	acc	prec	rec	roc_auc	avg. prec
TransH							
+VM	NB	0.765	0.667	0.687	0.864	0.598	0.679
-VM	NB	0.766	0.675	0.699	0.847	0.615	0.688
+VM	CART	0.713	0.675	0.801	0.642	0.686	0.740
-VM	CART	0.711	0.671	0.795	0.642	0.681	0.736
DistMult							
+ VM	NB	0.740	0.663	0.710	0.785	0.621	0.693
-VM	NB	0.678	0.611	0.706	0.653	0.596	0.679
+VM	RF	0.699	0.664	0.802	0.619	0.680	0.736
-VM	RF	0.704	0.663	0.788	0.636	0.673	0.729
+VM	SVM(RBF)	0.736	0.676	0.756	0.718	0.662	0.720
-VM	SVM(RBF)	0.746	0.680	0.745	0.746	0.657	0.715
+VM	MLP	0.725	0.689	0.816	0.652	0.702	0.751
-VM	MLP	0.744	0.703	0.813	0.686	0.710	0.755

ne Veränderungsmodell kaum unterscheiden. In den meisten Fällen unterscheiden sie sich um weniger als 0.01. Lediglich bei der Kombination von Distmult-Embeddings mit einem Veränderungsmodell und dem Naive-Bayes-Verfahren zeigen sich Unterschiede. Der Aspekt, der durch das Naive-Bayes-Verfahren wahrscheinlich abgedeckt wird, kann anhand von Abbildung 7.13 betrachtet werden. Die Linie repräsentiert die durchschnittliche Verteilung des Labels $\boxed{n}$ (also die Klasse $C_{nochange}$) über alle Veränderungsmodell-Klassen. Die Klasse *Renamed* scheint auf den ersten Blick gut zur Vorhersage geeignet zu sein, allerdings kommt die Klasse *Renamed* in dem Trainingsdatensatz nicht vor. Die Veränderungsmodell-Klasse *Superclass* wird etwas häufiger mit $\boxed{n}$ gelabelt. Dementsprechend kann dieser Zusammenhang gelernt werden. Allerdings sind die Ergebnisse insgesamt schlechter als bei anderen Kombinationen. Aus diesem Grund wurde dieses Verfahren zur Kombination von Embeddings und Veränderungsmodellen nicht weiter betrachtet.

7.4 Zusammenfassung der Ergebnisse

Im Rahmen dieses Kapitels wurden drei Untersuchungen vorgestellt:

1. Ein Anwendungsbeispiel aus dem Bereich der Interweaving Systems für einen regelbasier-

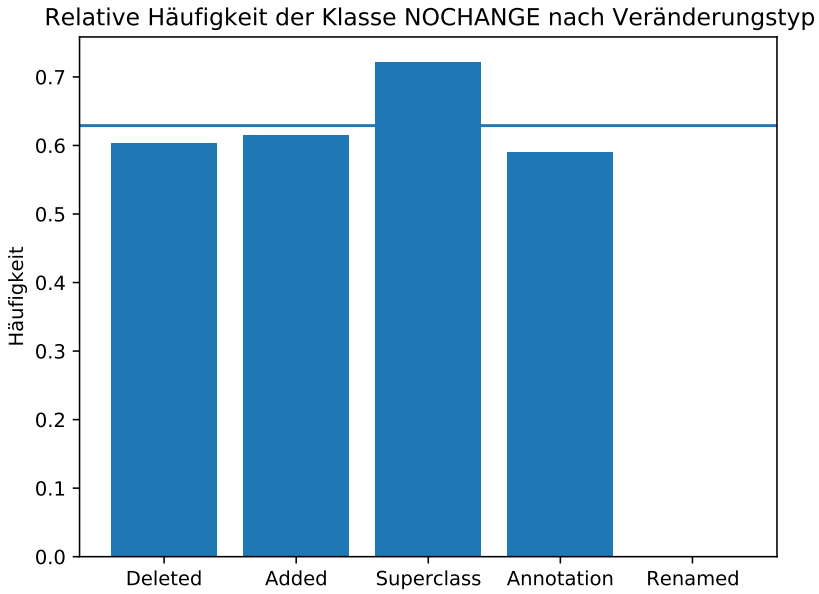

Abbildung 7.13: Häufigkeit der Klasse NOCHANGE ($C_{nochange}$) nach Klasse des Veränderungsmodells in den Trainingsdaten

ten Ansatz, der Inferenzen mit einbezieht (siehe Abschnitt 7.2),

2. Eine Evaluation des Embedding- und Graph-Convolution-basierten Ansatzes auf einem Datensatz aus dem Bereich der Biomedizinischen Ontologien (Abschnitt 7.3.5), und

3. Eine Evaluation der Kombination von Embeddings und eines Veränderungsmodells auf einem Teil des Biomedizinischen Datensatzes (Abschnitt 7.3.8).

Die wichtigsten Ergebnisse der ersten Untersuchung sind, dass sich ein Ansatz, der regelbasiert Inferenzen einbezieht, erfolgreich auf praxisnahe Beispiele anwenden lässt. Die Anwendbarkeit ist auch bei technischen Anwendungsfällen und bei beschränkter Rechenkapazität gegeben. In dem konkreten Anwendungsfall konnte gezeigt werden, dass der Ansatz in dem Beispiel, ähnlich wie ein modellbasierter, Ansatz helfen kann die benötigten Code-Zeilen einer Implementierung zu reduzieren. Dies wurde anhand eines domänenunabhänigen Ansatzes zur Verbindung von Interweaving Systems demonstriert.

Die zweite Untersuchung konnte zeigen, dass gelernte strukturelle Eigenschaften von Graphen Informationen beinhalten, die verwendet werden können, um Anpasungsbedarf an Alignments vorherzusagen. Sowohl Embeddings als auch spezielle Graph-Netzwerke wie Graph Convolutional Networks können für diese Aufgaben prinzipiell verwendet werden. Im Vergleich scheinen Graph Convolutional Networks insgesamt am besten geeignet zu sein, um Alignment-Änderungen vorherzusagen.

Hauptergebnis der dritten Untersuchung ist, dass der gewählte Ansatz zur Kombination von Embeddings und Veränderungsmodellen keinen erkennbaren Vorteil gegenüber der ausschließlichen Verwendung von Embeddings besitzt. Es konnten zwar Unterschiede zwischen der Verwendung von Embeddings und der Kombination aus Veränderungsmodell und Embeddings beobachtet werden, allerdings waren diese Unterschiede eher klein. Nur in einem Fall konnte die Verwendung eines Veränderungsmodells bessere Ergebnisse erzielen. In diesem Fall war die Differenz zur nächstbesten Performance ohne Veränderungsmodell nur sehr gering.

Kapitel 8

Diskussion

Anhand der vorgestellten Evaluation lassen sich einige Schlussfolgerungen ziehen. Dazu werden in diesem Kapitel die Ergebnisse der Arbeiten, die in Kapitel 7 vorgestellt wurden, diskutiert und bewertet. Zuerst werden die Ergebnisse zu den in Kapitel 4 vorgestellten Forschungsfragen auf einer abstrakten Ebene zusammengefasst (Abschnitt 8.1). Diese abstrakte Beantwortung der Forschungsfragen verbindet die Fragen mit den in der Arbeit untersuchten Ansätzen: Insgesamt wurden vier Ansätze vorgestellt, diese Ansätze werden im weiteren Verlauf der Diskussion detaillierter im Bezug auf ihre jeweiligen Vor- und Nachteile betrachtet (Abschnitt 8.2). Im Anschluss werden die Grenzen dieser Arbeit und der darin vorgestellten Ansätze und Evaluation dargestellt (Abschnitt 8.3). Am Ende der Diskussion werden einige allgemeinere Beobachtungen vorgestellt (Abschnitt 8.4).

8.1 Beantwortung der Forschungsfragen

In Abschnitt 4.2 wurden sechs Forschungsfragen formuliert. Zu jeder dieser Fragen wird an dieser Stelle die Frage selbst, die Methode, mit der diese Frage untersucht wurde, sowie die Ergebnisse kurz zusammengefasst:

RQ1 *Welche Methoden können für die Klassifikation, ob Veränderungen an den Ontologien Veränderungen am Alignment nach sich ziehen, angewendet werden?* Im Rahmen der Beantwortung dieser Forschungsfrage wurde das Problem als eine Fragestellung auf sich verändernden Graphen formuliert und vier Ansätze für diese Problemstellung vorgestellt: (1) ein Ansatz, der die Inferenzen bei der Veränderung von Ontologien betrachtet und mithilfe von Regeln auf Veränderungen reagiert; (2) ein Ansatz, der Embeddings und Verfahren des maschinellen Lernens verwendet, um Änderungen am Alignment vorherzusagen; (3) ein Ansatz, der diese Aufgabe mit Graph Convolutional Networks umsetzt

und (4) ein Ansatz, der ein Veränderungsmodell mit Embeddings kombiniert.

Für die Ansätze 1, 2 und 3 konnte in Anwendungsfällen gezeigt werden, dass die Ansätze in diesen Anwendungsfällen eingesetzt werden können und sinnvolle Ergebnisse erzeugen. Ansatz 4 konnte in dem dargestellten Anwendungsfall auch angewandt werden, es konnten allerdings keine Vorteile gegenüber Ansatz 2 festgestellt werden. Da Ansatz 4 eine Erweiterung von Ansatz 2 darstellt, ist die Erweiterung als nicht praktikabel zu bewerten (siehe RQ 6).

RQ2 *Welchen Effekt hat das Einbeziehen von Inferenzen in den Prozess zur Anpassung des Alignments an Veränderungen?* Hinsichtlich dieser Froschungsfrage wurde feststellt, dass Inferenzen in den Prozess mit einbezogen werden können. An einem Anwendungsbeispiel wurde gezeigt, dass das Einbeziehen von Inferenzen eine effiziente Modellierung erlaubt. In dem Anwendungsbeispiel konnte der Code, der für den domänenspezifischen Teil der Implementierung verwendet wurde, sehr klein gehalten werden. Diese kompaktere Implementierung kann auf die Einbeziehung der Inferenzen zurückgeführt werden, da ohne die Verwendung der Inferenzen eine explizite Modellierung der semantischen Grundlage notwendig gewesen wäre.

RQ3 *Welche Einschränkungen müssen für Ontologien und Alignments gelten, damit ein regelbasierter Ansatz angewandt werden kann?* Zur Beantwortung der dritten Forschungsfrage war es notwendig, einschränkende Annahmen zu formulieren. Dabei wurden die Einschränkungen so gewählt, dass der Ansatz in jedem Fall funktioniert. Einschränkungen wurden bezüglich der betrachteten Veränderungen, der Notation der Eingabedaten und der Verwendung von Reasoning-Algorithmen festgelegt.

Diese gewählten Einschränkungen sind teilweise enger, als für die Anwendung des Ansatzes zwingend nötig gewesen wären, da sie so gewählt wurden, dass der Ansatz sehr einfach angewendet werden kann. Es sind teilweise auch weitere Einschränkungen denkbar, diese wurden allerdings im Rahmen dieser Arbeit nicht untersucht. Für einige Einschränkungen ist allerdings klar, dass diese auch weiter gefasst werden können: zum Beispiel wurde in Abschnitt 5.1.3 festgelegt, dass nur Subclass-Reasoning verwendet wird. Diese Einschränkung wurde allerdings nur so gewählt, damit der Anwendungsfall einfach umsetzbar ist. Prinzipiell sind alle Arten von Reasoning verwendbar.

RQ4 *Ist es möglich die inhärente Graphstruktur von Alignments und Ontologien direkt mit einzubeziehen? Wenn ja, welche Verfahren können dazu verwendet werden?* Zur Untersuchung der vierten Forschungsfrage wurden mehrere Verfahren betrachtet, die die Graph-

struktur der Ontologien und des Alignments mit einbeziehen. Die Graphstruktur wird dabei durch mit Verfahren des maschinellen Lernens erzeugten Features repräsentiert.

Eine Kategorie der betrachteten Verfahren sind Graph Embeddings, die durch Vorhersagen von Kanten im Graph Features erzeugen. Ein weiteres betrachtetes Verfahren sind Graph Convolutional Networks. Bei diesen Verfahren wird die Graphstruktur im neuronalen Netzwerk selbst abgebildet. Diese Ansätze sind in der Lage, die Struktur der Ontologien zu betrachten, um Alignment-Veränderungen vorherzusagen.

RQ5 *Können moderne Methoden des maschinellen Lernens wie Graph Neural Networks und Knowledge Graph Embeddings zur Anpassung von Alignments an Veränderungen genutzt werden?* Hinsichtlich der fünften Forschungsfrage wurden sowohl Experimente mit Knowlege Graph Embeddings als auch mit Graph Neural Networks durchgeführt. Für beide Ansätze konnte gezeigt werden, dass bessere Ergebnisse erzielt werden können, als wenn Klassen ohne Betrachtung des konkreten Beispiels zugewiesen werden, zum Beispiel durch das Zuweisen der Klasse, die in der Mehrheit ist, oder durch zufälliges Zuweisen einer Klasse. Die Ergebnisse, die mit Knowledge Graph Embeddings und Graph Convolution erzeugt werden können, unterscheiden sich: während auf einem Datensatz Graph Embeddings und Graph Convolutional Networks sehr ähnliche Ergebnisse erzeugt werden konnten, konnten auf einem anderen Datensatz mit Graph Convolutioneal Networks die besten Ergebnisse erzeugt werden.

RQ6 *Können diese Methoden des maschinellen Lernens auch mit Veränderungsmodellen verbunden werden? Welche Vorteile bietet ein solcher Ansatz?* Im Rahmen der Beantwortung der letzten Forschungsfrage wurde ein Verfahren zur Kombination von Graph Embeddings mit einem Veränderungsmodell vorgestellt, bei dem ein Feature-Vektor erstellt wird, der aus Embedding und Veränderungsrepräsentation besteht. Bei der Evaluation dieses Verfahrens konnte allerdings kein Vorteil gegenüber der ausschließlichen Verwendung von Embeddings festgestellt werden (Siehe auch RQ1).

8.2 Bewertung der Ansätze

Bei der Betrachtung der Ansätze können diese aufgeteilt werden in Ansätze, die manuell erstellte Regeln einsetzen und solche, die maschinelles Lernen verwenden. In die erste Kategorie fällt der regelbasierte Ansatz zur Einbeziehung von Inferenzen, in die zweite Kategorie alle anderen Ansätze. Für beide Kategorien ergeben sich direkte Vor- und Nachteile. Ein wichtiger Aspekt ist die Nachvollziehbarkeit der Entscheidungen, die ein Verfahren trifft. Regeln kön-

nen zum Beispiel von Experten betrachtet und auf ihre grundsätzliche Plausibilität untersucht werden. Diese ist bei Methoden, die auf neuronalen Netzen aufbauen grundsätzlich schwierig. Es gibt einige Ansätze, die versuchen, Methoden des maschinellen Lernens erklärbar zu machen [Hol18], diese sind aber auf die in dieser Arbeit verwendeten Verfahren nicht anwendbar, vor allem, weil die in der Arbeit eingesetzten Ansätze keine Features verwenden, die bei einer getrennten Betrachtung sinnvoll interpretiert werden können. Ein weiterer Aspekt, auf den hier Rücksicht genommen werden sollte, sind die Erkenntnisse, die aus der Anwendung der Verfahren gewonnen werden können: werden Regeln erstellt, können prinzipielle Eigenschaften der Domäne entdeckt werden, die direkt auch in Regeln abgebildet sind. Dabei wird auch das Verständnis der Problemdomäne verbessert. Bei Verfahren des maschinellen Lernens, bei denen man den Entscheidungsprozess nicht direkt nachvollziehen kann, können höchstens abstrakte Schlussfolgerungen über die Domäne gezogen werden, zum Beispiel welche strukturellen Eigenschaften scheinbar wichtig für ein Klassifikationsverfahren sind und welche Verfahren in der Domäne angewendet werden können und welche nicht.

Ein weiterer prinzipieller Unterschied ist die Anpassung an Änderungen im Veränderungsprozess. Wenn sich die grundsätzliche Art, wie auf Veränderungen reagiert werden muss ändert, kann ein Verfahren, das auf maschinellem Lernen aufbaut, einfach auf weiteren Daten trainiert werden. Bei einem Ansatz mit fest vorgegebenen Regeln müssen die Regeln, die die Reaktion auf Veränderungen bestimmen, vollständig überarbeitet werden, was einen großen Aufwand bedeutet. Ein Ansatz, der maschinelles Lernen verwendet, benötigt allerdings weitere Trainingsdaten, die zumindest teilweise manuell annotiert werden müssen. Auch in diesem Fall ist also manueller Aufwand notwendig.

Auch wenn der Ansatz auf neue Daten angewandt werden muss, zeigen sich Unterschiede zwischen den grundsätzlichen Herangehensweisen: Ansätze, die maschinelles Lernen einsetzen, brauchen lediglich einen annotierten Trainingsdatensatz. Im Gegensatz dazu muss bei der Anwendung eines regelbasierten Ansatzes in einer neuen Domäne ein komplett neuer Regelsatz erzeugt werden, was nur möglich ist, wenn die Art der Veränderungen vollständig verstanden werden.

8.2.1 Vergleich der Vorbedingungen zur Verwendung der Ansätze

Bei der Vorstellung der verschiedenen Ansätze in Kapitel 5 werden jeweils Vorbedingungen zur Anwendbarkeit der Ansätze vorgestellt. Diese Vorbedingungen sind ein erstes Kriterium zum Vergleich der Ansätze. Diese Vorbedingungen unterscheiden sich teilweise sehr. Eine Übersicht über diese Vorbedingungen ist in Tabelle 8.1 dargestellt. In der Tabelle bezeichnet „Regeln und Inf.“ den Ansatz, der Inferenzen regelbasiert in den Veränderungsprozess mit einbezieht; „Em-

Tabelle 8.1: Vergleich der Vorbedingungen zur Verwendung der Ansätze

	Nicht Lernend	Neuronale Netze		
	Regeln und Inf.	Embeddings	Embeddings + VM	RGCN
Eingabedaten	✓	✓	✓	✓
Trainingsdaten	×	✓	✓	✓
Embedding	×	✓	✓	×
Klassifikationsverfahren	×	✓	✓	×
Veränderungsmodell	✓	×	✓	×
Einschränkung der Eingabedaten	(✓)	×	×	×
Vorgabe von Berechnungsvorschriften	✓	×	×	×

beddings“ bezeichnet die Kombination von Graph Embeddings und Klassifikationsverfahren, „Embeddings + VM“ bezeichnet die Erweiterung des Ansatzes zur Kombination von Embeddings und Klassifikationsverfahren um eine Veränderungsmodell und „RGCN“ bezeichnet den Einsatz von Relational Graph Convolutional Networks.

Die erste Zeile – „Eingabedaten“ – bezieht sich darauf, dass die Verfahren als Eingabedaten die vollständigen Ontologien und das Alignment zwischen diesen sowie alle Veränderungen benötigen. „Trainingsdaten“ beschreibt, in wie fern zum Training Daten mit korrekten Lösungen vorhanden sein müssen. „Embedding“ und „Klassifikationsverfahren“ repräsentieren, ob ein solches Verfahren ausgewählt und implementiert werden muss, um den Ansatz einsetzen zu können. Mit „Veränderungsmodell“ wird beschrieben, ob ein Verfahren eine Klassifizierung von Veränderungen durch ein Veränderungsmodell benötigt. „Einschränkung der Eingabedaten“ repräsentiert, ob es Einschränkungen gibt, die die Eingabedaten betreffen. Hier ist der Eintrag bei Regeln und Inferenz mit Klammern versehen, da die in dieser Arbeit festgelegten Einschränkungen möglicherweise aufgeweicht werden könnten. Die letzte Zeile – „Vorgabe von Berechnungsvorschriften“ – bezeichnet, ob für das Verfahren einige Berechnungsvorschriften je nach Domäne definiert werden müssen.

Beim Betrachten der Tabelle fällt auf, dass die Vorbedingungen für die lernenden und nicht lernenden Verfahren sich stark unterscheiden. Das Verfahren mit den wenigsten Vorbedingungen ist der Einsatz von Relational Graph Convolutional Networks.

8.2.2 Regelbasierter Ansatz mit Einbeziehung von Inferenzen

Für den in Abschnitt 5.1 vorgestellten und in Abschnitt 7.2 evaluierten Ansatz konnten einige Vorteile aufgezeigt werden: Die Modellierung des Anwendungsfalls konnte sehr kompakt gestaltet werden. In dem konkreten Beispiel konnte die Regel für die Reaktion auf Veränderungen in Abschnitt 7.2.3 sehr kompakt notiert werden. Die Funktion zur Bestimmung des Änderungsbedarfs müsste ohne die Verwendung von Inferenzen eine Betrachtung der grundlegenden Semantik auf andere Art mit einbeziehen. Das würde bedeuten, das die Unterklassen-Semantik Teil der Regel sein müsste. Wenn es wie im dargestellten Beispiel nur wenige Regeln gibt, ist das zwar ein zusätzlicher Aufwand, dieser ist allerdings noch größer, wenn es mehr Regeln gibt und noch weitere Arten von Semantik abgebildet werden müssen. Insgesamt macht das Einbeziehen von Inferenzen eine Trennung von solchen Inferenzregeln und der Beschreibung der durch die Domäne bestimmten Veränderungsregeln möglich.

Die auf einen Aspekt fokussiertere und somit kompaktere Beschreibung kann auch helfen, eine bessere Übersicht über die domänenspezifischen Regeln zu erhalten, da diese nicht mit den Inferenz-Regeln gemeinsam betrachtet werden müssen. Diese Trennung von unterschiedlichen Aspekten, also eine *Seperation of Concerns*, kann außerdem zur Vermeidung von Fehlern führen und Systeme einfacher wartbar machen. Auch die Benutzbarkeit der Modelle kann davon profitieren: dadurch, dass verschiedene Aspekte an verschiedenen Stellen modelliert sind, kann sich beim Erzeugen der Modelle auf einen Aspekt fokussiert werden. Dadurch kann man sich bei der Modellierung der Regeln auf diesen Domänen-Aspekt konzentrieren und wird weniger durch andere Aspekte (zum Beispiel technische Aspekte) abgelenkt.

Ein wichtiger Nachteil dieses Ansatzes ist, dass der Aufwand zum Berechnen von Inferenzen sehr unterschiedlich sein kann: Das Berechnen der Unterklassen-Inferenzen kann mit einem Algorithmus zur Berechnung der transitiven Hülle über die Unterklassen-Statements durchgeführt werden. Dazu kann der Algorithmus von Floyd-Warshall angewandt werden [Flo62; War62]. Die Laufzeitkomplexität des Floyd-Warshall-Algorithmus ist bei einem gegebenen Graphen $G(V, E, s, t, \Lambda)$ für die Anwendung auf dem gesamten Graphen $\mathcal{O}(|V|^3)$. Die Komplexität für Profile der verbreiteten Ontologiesprache OWL 2 reicht von Problemen in der Klasse NL (logarithmische Speicherkomplexität mit nichtdeterministischen Turingmaschinen) bis zu NEXPTIME (exponenzieller Zeitaufwand mit nichtdeterministischen Turingmaschinen), einige Profile sind sogar unentscheidbar [Mot+12]. Dabei bezieht sich die Komplexität der OWL-Profile nicht wie beim Floyd-Warshall-Algorithmus auf die Zahl der Knoten (V), sondern auf die Zahl der Kanten im Graph ($|E|$), was den Unterschied zwischen der Komplexität für Subklasseninferenz und vollständiger OWL-Inferenz nochmal verstärkt. Beispielsweise war im Jahr 2014 $|V|$ für die populären Wissensgraphen DBpedia 25 Millionen, für Wikidata etwa 20 Millionen und

für YAGO3 300 Millionen, während $|E|$ für DBPedia 400 Millionen, für Wikidata 100 Millionen und für YAGO3 1 Milliarde [FR15] war. Aus diesem Grund wendet der in dieser Arbeit vorgestellte Ansatz die Berechnung von Inferenzen nur auf einem Ausschnitt der Ontologien an, dem entsprechend sind die Eingabedaten für das Reasoning deutlich kleiner.

Ein weiterer Vorteil dieses Ansatzes ist, dass er bereits auf kleineren Daten angewendet werden kann. Dann sind auch die Auswirkungen der Laufzeitkomplexität weniger ausschlaggebend. Außerdem ist es wahrscheinlich, auf weniger Daten einfacher Regeln finden zu können, da die Komplexität möglicherweise geringer ist. Die Anwendung auf kleinen Daten ist insbesondere möglich, da keine Trainingsdaten gebraucht werden, um den Ansatz anwenden zu können.

8.2.3 Embedding-basierter Ansatz

Der in Abschnitt 5.2 vorgestellte Ansatz und die in Abschnitt 7.3 beschriebene Evaluation weisen auf einige Vorteile des Ansatzes hin: Die Verwendung von Embeddings ermöglicht die Abbildung der Graphstruktur und die Verwendung von Klassifikationsalgorithmen. Sowohl das Erzeugen von Embeddings als auch die Klassifikationsalgorithmen sind in der Forschung bereits detailliert betrachtet worden und es stehen Libraries zur Verfügung, mit deren Hilfe diese Verfahren verhältnismäßig einfach umgesetzt werden können. Dies reduziert die Fehleranfälligkeit der Umsetzung und erlaubt es, schneller zu einer funktionierenden Lösung zu kommen. Außerdem stehen in beiden Fällen viele Alternativen zur Verfügung, die unterschiedliche Stärken und Schwächen haben.

Ein weiterer Vorteil ist, dass kein aufwändiges Feature Engineering notwendig ist, um die Graphen zu repräsentieren, da die Features automatisch von Graph-Embedding-Verfahren erzeugt werden. Da passende Features je nach Anwendungsfall sehr unterschiedlich sein können, ist anzunehmen, dass sich das Verfahren einfacher auf neue Anwendungsbereiche übertragen lässt als ein Verfahren, welches manuelles Feature-Engineering verwendet.

Eine Schwierigkeit dieses Ansatzes ist der Zeit- und Speicherplatzaufwand beim Lernen der Embeddings: das Erzeugen von Graph-Embeddings ist mit einem hohen Ressourceneinsatz verbunden. Da für jeden Knoten ein Vektor erzeugt werden muss, werden für das Embedding eines Graphen $G(V, E, s, t, \Lambda)$ immer mindestens $|V| \times d \times 4$ Bytes an Speicherplatz benötigt (d repräsentiert die Dimensionalität der Embeddings). Bei einem Beispiel wie dem in Abschnitt 7.3.1 beschriebenen Datensatz FMA-NCI führt das bei $d = 500$ zu einem Speicherplatzbedarf von 4 GB, für SNOMED-FMA sogar zu 8 GB.

Die Laufzeitkomplexität des Embedding-Verfahrens TransE ist mit $\mathcal{O}(|E| \times d)$ für jede Trainingsepoche linear in der Zahl der Kanten. Große Wissensgraphen verfügen allerdings über sehr viele Kanten. In einer Betrachtung der Laufzeit von solchen Embedding-Ansätzen wurden

TransE-Embeddings für den gesamten Freebase-Datensatz gelernt. Beim Einsatz von zwei Intel Xeon E5-2640 CPUs (mit jeweils 10 Kernen und × 2 Hyperthreading, und einer Taktfrequenz von 2,4 GHz) mit 128 GB Arbeitsspeicher hätte dies zu einer Trainingszeit von 357 Tagen [Zha+17] geführt. Für andere Modelle wie DistMult und Complex ist die Laufzeit ebenfalls linear in der Anzahl der Kanten [KP18]. Dem entsprechend ist auch hier für große Datensätze mit erheblichem Zeitaufwand zu rechnen, alleine aus dem Grund, dass moderne Wissensgraphen so viele Kanten enthalten können.

Auch in der hier durchgeführten Evaluation konnten nicht alle Embedding-Ansätze auf allen Datensätzen evaluiert werden, da das betrachtete Beispiel für den Einsatz von RDF2Vec auf der verfügbaren Hardware zu groß war. Bereits beim Erzeugen der Random Walks wären mehrere Monate Rechenzeit und mehrere Terabyte Speicherplatz benötigt worden, weshalb das Experiment abgebrochen wurde. Dabei wird der Großteil der berechneten Embeddings nicht benötigt. Nur ein sehr kleiner Teil der Knoten wird verändert und nur die Embeddings dieser Knoten werden tatsächlich bei der Klassifikation verwendet. In weiteren Arbeiten könnte untersucht werden, wie solche Embeddings effizienter erzeugt werden können, wenn als Ergebnis nur die Embeddings von wenigen Knoten relevant sind. Hier kann beispielsweise eine Sampling-Strategie angewendet werden, die für weniger relevante Knoten nur wenige oder keine Beispiele beim Training betrachtet als für relevante Knoten. Dafür müssten allerdings die relevanten Knoten beim Erstellen der Embeddings bereits bekannt sein.

Ein weiterer möglicher Nachteil des Ansatzes ist die Trennung des Trainings der Repräsentation und der Klassifikation: Das Embedding wird so erzeugt, dass es die für das Embedding definierte Zielsetzung erfüllt, nämlich die Vorhersage von weiteren Kanten oder das Vorhersagen von weiteren Stellen in einem Random Walk. Das ist eine deutlich andere Aufgabenstellung als das Vorhersagen von Alignment-Änderungen für veränderte Knoten. Es ist zwar möglich die vortrainierten Embeddings zu „fine-tunen“, dies ist beispielsweise für Anwendungen auf natürlicher Sprache üblich [ZW15]. Damit wäre man allerdings auf ein neuronales Netz als Klassifikationsmodell festgelegt. Zukünftige Arbeiten können untersuchen, wie Fine-Tuning von Embeddings in diesem Kontext sinnvoll eingesetzt werden kann. Ein weiterer Nachteil ist, dass ein solches Modell mit zwei getrennten Schritten schwer zu optimieren ist. Um zu einer optimalen Konfiguration zu kommen, müsste man die Hyperparameter des Embedding-Verfahrens und die Hyperparameter des Klassifikationsverfahrens zusammen optimieren, wobei auch verschiedene Kombinationen von Embedding und Klassifikationsverfahren evaluiert werden müssen. Das führt zu sehr vielen Kombinationen und damit zu einem besonders aufwendigen Training und Evaluation.

8.2.4 Kombination von maschinellem Lernen und Veränderungsmodellen

Für den in Abschnitt 5.3.2 vorgestellten und in Abschnitt 7.3.8 evaluierten Ansatz konnten anhand der erzeugten Ergebnisse keine Vorteile gegenüber den anderen Ansätzen gezeigt werden. Bei allen anderen Ansätzen konnten unter der Verwendung von weniger Informationen bessere oder vergleichbare Ergebnisse erzielt werden. Dafür kann es verschiedene Gründe geben: Es ist denkbar, dass die gewählte Methodik zur Kombination von Veränderungsmodell und Embeddings nicht geeignet sind, um etwas Relevantes zu lernen. Es ist denkbar, dass bei der gewählten Kombinationsmethode nur schwer identifiziert werden kann, welche Features einen Einfluss auf das Ergebnis haben.

Eine weitere denkbare Ursache ist, dass das Veränderungsmodell nicht geeignet ist, um Vorhersagen zu treffen. Darauf deutet die in Abbildung 7.13 dargestellte Verteilung der Klassen nach Typ im Veränderungsmodell hin: abgesehen von der Klasse, die die Veränderung der Oberklasse abbildet, sind die Verteilungen der Klassen fast gleich. Dementsprechend kann es auch schwierig sein, anhand dieser Veränderungsklassen Vorhersagen zu treffen. Es können auch weitere Veränderungsmodelle betrachtet werden. Beispielsweise wurden neben dem Veränderungsmodell von owl-diff, welches in dieser Arbeit verwendet wird, in der Literatur mehrere Veränderungsmodelle für Ontologien vorgestellt [Har+13], [NM04], [Sto+02]. Welche dieser Veränderungsmodelle für eine Verbindung mit Embeddings geeignet sind, müsste noch untersucht werden.

Ein weiterer möglicher Grund für die Schwierigkeiten beim Einbeziehen des Veränderungsmodells kann auch die Formulierung der Problemstellung sein: Die Problemstellung wurde so gewählt, dass Vorhersagen über die Nachbarschaft von veränderten Knoten getroffen werden sollen. Wenn mehrere Knoten in der Nachbarschaft verändert wurden, ist die Art der Veränderung nicht mehr so bedeutend, wie wenn nur ein Knoten verändert wird. Dies kann auch ein Grund sein, warum gegenüber der ausschließlichen Verwendung von Embeddings kein Vorteil gesehen werden kann.

8.2.5 Graph Convolutional Networks

Aus der in Abschnitt 7.3 beschriebenen Evaluation des Ansatzes, der auf Graph Convolutional Networks aufbaut (siehe Abschnitt 5.4) lassen sich einige Schlussfolgerungen ableiten: Die Leistung der Graph Convolutional Networks war im Rahmen der dargestellten Evaluation mindestens vergleichbar mit den Ergebnissen der Embedding-basierten Lösung. Ein Grund für das Übertreffen des Embedding-Basierten Ansatzes kann die Anpassbarkeit der Repräsentation

sein: anstatt dem Verwenden einer ein mal gewählten Repräsentation wird die Repräsentation für die vorliegende Aufgabe spezifisch gelernt. Außerdem verringert sich durch die Verwendung eines Verfahrens für Repräsentation und Klassifikation die Anzahl der Hyperparameter, was eine umfassendere Evaluation erleichtert.

Außerdem ist das Laufzeitverhalten dieses Ansatzes anders: Die Komplexität für eine Trainingsepoche ist für einen Graphen $G(V, E, s, t, \Lambda)$ $\mathcal{O}(|E|)$ [Wu+20]. Diese Laufzeit gilt für das Lernen der Repräsentation und der Klassifikation. Beim Ansatz, der Embeddings einsetzt, müssen diese Aufwände getrennt betrachtet werden. Alleine für das Lernen der Embeddings wird also mehr Zeit benötigt als für die gesamte Klassifikation.

Allerdings wird bei der Anwendung von Relational Graph Convolutional Networks deutlich mehr Arbeitsspeicher benötigt als für Graph Embeddings. Der Speicherbedarf ergibt sich direkt aus der Modellarchitektur: Für jedes Layer des GCN werden $|V| \times d$ Parameter benötigt, also genau so viele wie für das Erstellen der Embeddings. Allerdings wird dieser Speicherplatz für jedes Layer benötigt, nicht nur ein mal wie beim Erstellen der Embeddings. Für sehr große Datensätze wie DBPedia oder Wikidata kann der Ansatz so also gar nicht oder nur unter dem Einsatz von sehr viel Arbeitsspeicher angewandt werden. Der Arbeitsspeicherbedarf eines einzelnen Layers schränkt auch die Tiefe des Netzes für große Graphen ein. Dadurch wird auch beeinflusst, wie weit Signale zwischen den einzelnen Knoten des Graphen ausgetauscht werden, da zwischen zwei Layern nur direkt verbundene Knoten Informationen austauschen können. Für Aufgabenstellungen, bei denen vor allem lokal begrenzte Phänomene betrachtet werden, stellt dies vermutlich kein großes Problem dar, da die Betrachtung der mittelbaren Nachbarschaft eines Graphknoten immer noch möglich ist. Sollen allerdings Informationen von weit entfernten Bereichen des Graphen berücksichtigt werden, kann dieser Ansatz nur schwer verwendet werden. Dies ist eine zentrale Eigenschaft von Graph Convolutional Networks: Aktivität in der unmittelbaren Umgebung der Graph-Knoten wird am meisten Bedeutung beigemessen.

8.2.6 Zusammenfassung der Vor- und Nachteile

Eine tabellarische Übersicht des in den vorherigen Abschnitten beschriebenen Vergleichs ist in Tabelle 8.2 dargestellt. In der Tabelle werden die Ansätze genau so bezeichnet wie auch beim Verleich der Vorbedingungen in Tabelle 8.1.

Die ersten Zeilen des Vergleichs beziehen sich in erster Linie auf den Vergleich zwischen lernenden und nicht lernenden Methoden. Eine dieser Zeilen ist die „Erklärbarkeit“. In dieser Zeile wird dokumentiert, ob erkennbar ist, wie das Verfahren zu einer Entscheidung gekommen ist. In der Zeile „einfache Übertragbarkeit auf eine neue Domäne“, wird festgehalten, ob eine Übertragung eines Ansatzes auf eine andere Domäne potenziell einfach möglich ist. Die Zeile „Vollst.

Tabelle 8.2: Zusammenfassung der Vor- und Nachteile

	Nicht Lernend	Neuronale Netze		
	Regeln und Inf.	Embeddings	Embeddings + VM	RGCN
Erklärbarkeit	✓	×	×	×
Einfache Übertragbarkeit auf neue Domäne	×	✓	✓	✓
Vollst. Verständnis der Domäne notwendig	✓	×	×	×
Trainingsdaten benötigt	×	✓	✓	✓
Basis für MDE	✓	(×)	(×)	(×)
Komplexität	$\mathcal{O}(\|V\|^3)$-NEXPTIME	$\|E\|$ für Embedding, Speicherbedarf $\|V\| * d$	Wie Embeddings, abhängig von VM	Laufzeit $\|E\|$, Speicherbedarf $\|V\| * d$ je Layer
Anpassbarkeit der Repräsentation an Task	n.a.	×	×	✓
Genauigkeit	Vollständige Modellierung möglich	Am besten auf kleinerem Datensatz	keine Vorteile ggü. anderen	Am besten auf beiden Datensätzen

Verständnis der Domäne notwendig“ beschreibt, ob jede Gesetzmäßigkeit der Domäne bekannt sein muss, um diese Gesetzmäßigkeit abbilden zu können. Eine weitere Zeile der Tabelle ist mit „Trainingsdaten benötigt“ gekennzeichnet. Diese Zeile stellt dar, ob annotierte Trainingsdaten benötigt werden, um den Ansatz einsetzen zu können. Als weitere Zeile stellt „Basis für MDE“ dar, ob sich der Ansatz für die Modellbasierte Entwicklung von Softwaresystemen eignet. Die mit Klammern versehenen Zellen symbolisieren, dass hier noch keine Untersuchung zu diesem Aspekt vorgenommen wurde.

Die übrigen Zeilen unterscheiden sich für jeden Ansatz. Mit „Komplexität“ wird die Laufzeit und der Speicherbedarf des ausschlaggebenden Teils der jeweiligen Ansätze bezeichnet. Die Zeile „Anpassbarkeit der Repräsentation an den Task“ betrifft nur die auf maschinellem Lernen aufbauenden Ansätze. Hier wird verglichen, ob die Repräsentation der Eingabedaten an die zu betrachtende Problemstellung angepasst werden kann. Die letzte Zeile betrachtet die „Genauigkeit“ der Ansätze. Während der Ansatz zur Regelbasierten Betrachtung von Inferenzen theoretisch eine beliebige Genauigkeit erreichen kann (Regeln könnten beliebig genau gefasst werden) werden die anderen Ansätze anhand ihrer Performanz auf den verschiedenen Datensätzen verglichen.

8.3 Grenzen der Arbeit

Hinsichtlich der Grenzen dieser Arbeit lassen sich mehrere Aussagen treffen. Diese Grenzen werden nach den durchgeführten Untersuchungen gegliedert, da je nach Untersuchung unterschiedliche Grenzen vorliegen.

In der Untersuchung des Ansatzes, der Inferenzen und regelbasierte Reaktion auf Veränderungen verbindet, wurden Einschränkungen vorgestellt (siehe Abschnitte 5.1 und 7.2), die zeigen, unter welchen Bedingungen der Ansatz in jedem Fall angewendet werden kann. Eine der Einschränkungen, die vorgestellt wurde, war, dass eine Funktion ρ existieren muss, die beschreibt, wie gut ein Knoten einen anderen ersetzen kann. Außerdem wird angenommen, dass wenn ein Alignment-Statement geändert wird, das korrekte neue Statement zwei Knoten mit dem maximalen Funktionswert für ρ verbindet. Es ist unklar, wie stark diese Annahme den Ansatz in der Praxis einschränkt. Für viele andere Einschränkungen kann, wie bereits im vorherigen Ansatz diskutiert, eine weniger strenge Alternativen gefunden werden, allerdings kann so nicht bestimmt werden, unter welchen Bedingungen der Ansatz nicht eingesetzt werden kann. Diese Fragestellung beantwortet diese Arbeit nicht.

Eine weitere Frage, die noch offen bleibt, sind die Vor- und Nachteile des Ansatzes bei einer Anwendung in der Praxis. Der Ansatz wurde der an einem praxisnahen Anwendungsbeispiel

vorgestellt. Es gibt einige Gründe, die dafür sprechen, dass der Ansatz Vorteile in der Praxis mit sich bringt, jedoch müssten an noch praxisnäheren Beispielen Vergleiche mit anderen Ansätzen vorgenommen werden, um zu Schlussfolgerungen zu kommen, die auch für die Praxis aussagekräftig sind. Mit den aktuell durchgeführten Experimenten lassen sich noch keine Aussagen dazu treffen. Insbesondere die Ausdrucksfähigkeit der Modellierung unter Berücksichtigung der Laufzeit müsste anhand von realen Beispielen untersucht werden. Für letzteren Aspekt wäre eine umfangreiche Studie von existierenden Modellen und Anwendungen notwendig.

Für alle Ansätze, die auf maschinellem Lernen aufbauen, sind die hier vorgestellten Ergebnisse auf die untersuchten Datensätze beschränkt. Die unterschiedliche Leistung der Modelle ist auf die Eigenschaften des Datensatzes und die darin vorkommenden Veränderungen beschränkt, auf anderen Datensätzen können die Ansätze auch ganz andere Ergebnisse erzeugen. Es gibt allerdings einige im Modell bedingte Hinweise, warum insbesondere der Graph-Convolution-Basierte Ansatz besser abschneidet als die anderen Ansätze, diese wurden im vorherigen Abschnitt diskutiert. Es ist also nicht unwahrscheinlich, dass einige Ergebnisse auf anderen Datensätzen ähnlich sein würden. Dies ist allerdings nur schwer zu überprüfen, da es zur Zeit noch sehr schwierig ist, Datensätze zu finden, auf die die Vorbedingungen zur Anwendbarkeit der Ansätze zutreffen.

Insbesondere für den Ansatz, der Embeddings und typische Klassifikationsverfahren kombiniert, konnten nicht alle Hyperparameter umfassend betrachtet werden. Aufgrund der großen Anzahl an Hyperparametern ist eine vollständigere Betrachtung nicht möglich gewesen. Dem entsprechend ist es durchaus denkbar, dass es eine Kombination aus Klassifikationsverfahren und Embedding gibt, die deutlich besser abschneidet, als die getesteten Kombinationen. Eine systematische Untersuchung wie zum Beispiel eine Rastersuche [Chi17] würde sehr viel Zeit in Anspruch nehmen, aus diesem Grund wurde sie im Rahmen dieser Arbeit auch nicht durchgeführt, da dies den Rahmen der Arbeit gesprengt hätte.

Auch für die Ergebnisse der Evaluation der Kombination aus Embedding-basierter Klassifikation und Veränderungsmodell (siehe Abschnitte 5.3 und 7.3.8) gibt es einige Einschränkungen: Dieser Ansatz wurde nur auf einem kleineren Datensatz evaluiert, dem entsprechend ist die Evaluation weniger aussagekräftig. Diese kleinere Evaluation sollte aber nicht unbedingt auf einem größeren Datensatz wiederholt werden: die Unterschiede zwischen der vorgestellten Erweiterung und der ausschließlichen Verwendung von Embeddings waren so gering, dass es nicht sinnvoll erscheint, eine weitere Evaluation durchzuführen. Es ist zwar theoretisch denkbar, dass auf einem größeren Datensatz bessere Ergebnisse erzielt werden können, das ist aber äußerst unwahrscheinlich.

Eine weitere Einschränkung ist die Art der Kombination von Embeddings und Verände-

rungsmodell und das gewählte Veränderungsmodell. Zur Art der Kombination von Embeddings und Veränderungsmodell wurden keine weiteren Experimente durchgeführt. Weitere Experimente schienen nicht sinnvoll, da das gewählte Veränderungsmodell, wie in Abschnitt 8.2.4, keine sinnvollen Vorhersagen erlaubt. Zum Veränderungsmodell selbst können weitere Untersuchungen durchgeführt werden. Hier müsste untersucht werden, welches Veränderungsmodell geeignet währe.

Bei der Interpretation der Ergebnisse zu Relational Graph Convolutional Networks (siehe Abschnitte 5.4 und 7.3) müssen neben den allgemeinen Einschränkungen, die für alle Ansätze gelten, die maschinelles Lernen einsetzen, einige weitere Einschränkungen betrachtet werden: Bei Relational Graph Convolutional Networks wurde nur eine Alternative von Graph Networks betrachtet, es sind noch weitere Alternativen denkbar. Ein Beispiel für ein Graph-Network, das vielversprechend eingesetzt werden kann, sind Relational Graph Attention Networks [Bus+19]. Für diese Netzwerke konnte allerdings erst 2019 eine Version vorgeschlagen werden, die vergleichbare Ergebnisse zu Relational Graph Convolutional Networks erzeugen kann, darum wurde diese Alternative nicht mehr in dieser Arbeit betrachtet.

8.4 Weitere Schlussfolgerungen

Neben den direkten, auf die konkreten Ansätze bezogenen Schlussfolgerungen ergeben sich aus der Arbeit noch weitere, abstraktere Beobachtungen. Interessanterweise konnten viele der Problemstellungen über Betrachtungen der direkten Nachbarschaft von Knoten behandelt werden. Viele der verwendeten Ansätze sind darauf auch ausgerichtet: Embeddings bilden Knoten mit ähnlichen Nachbarschaften an ähnlichen Orten im Vektorraum ab. Relational Graph Convolutional Networks übertragen von einem Layer zum nächsten immer nur die Signale der Knoten, die direkt miteinander verbunden sind. Auch der Ansatz zur Einbeziehung von Inferenzen arbeitet mit einem Nachbarschaftskonzept. Dies ist nicht unbedingt intuitiv: in Ontologien können auch die Strukturen in entfernten Oberklassen einen große Wirkung zeigen. In den meisten Graph-Repräsentationen scheint dies allerdings keinen Einfluss zu haben und auch in der direkten Nachbarschaft beobachtbar zu sein.

Eine weitere interessante Beobachtung ist, dass alleine aus der Struktur Aussagen über die Veränderungen von Alignments getroffen werden können. Es scheint strukturelle Eigenschaften zu geben, die die Stabilität in Bezug auf Alignments repräsentieren. Diese strukturellen Eigenschaften können anscheinend sowohl durch Embeddings als auch durch Graph Convolutional Networks erkannt und repräsentiert werden.

Sehr ähnliche Ergebnisse wurden auch bei der Betrachtung von natürlicher Sprache erziehlt.

Auch hier werden bei Ansätzen zur Repräsentierung von Sprache vor allem die Nachbarschft bei der Verwendung der Begriffe verwendet. So wird die Repräsentation bei Word2Vec so trainiert, dass zu einem Wort die Nachbarwörter, in denen es steht vorhergesagt werden sollen [Mik+13]. Ähnlich arbeiten modernere Modelle wie GPT-3, bei dem ebenfalls versucht wird, kleine Abschnitte von Text zu bestehendem Text vorherzusagen [Bro+20]. Außerdem verwenden diese Ansätze kein Vorwissen über die Sprache. Das heißt auch bei der Anwendung auf natürlicher Sprache wird nur die Struktur in der Verwendung untersucht. Da auch Graphen eine Sprache bilden, ist es nicht überraschend, dass ähnliche Techniken verwendet werden können, um Entitäten aus diesen Sprachen (also Knoten beziehungsweise Wörter) zu repräsentieren.

Eine weitere Beobachtung, die beim Betrachten der Arbeit aufkommt, ist die Bedeutung von Inferenzen in Ontologien und die damit einhergehneden Herausforderungen. Selbst wenn, wie im Falle der Evaluation in dieser Arbeit, nur Subclass-Reasoning verwendet wird, ist die Komplexität immer noch $\mathcal{O}(|V|^3)$. Zwar konnte in der Arbeit ein Weg gefunden werden, um mit dieser Komplexität umzugehen, allerdings ist trotzdem unklar, wie gut dies auf größeren Datensätzen funktionieren kann. Auf größeren Datensätzen sind die Knoten einer Ontologie häufig mit vielen Knoten verbunden, was auch bei dem in dieser Arbeit gewählten Ansatz für große Eingaben für das Reasoning führen würde. Damit würde auch die Laufzeit bei Anwendung der in dieser Arbeit vorgestellten Methode sehr lang werden. Dem entsprechend ist es grundsätzlich schwierig, Ansätze für große Wissensgraphen auf Inferenzen aufzubauen, wenn die Ontologie nicht auf kleinere Eingaben für die Berechnung der Inferenzen reduziert werden kann.

Kapitel 9

Fazit

Bei der Dokumentation von Wissen können Ontologien eingesetzt werden, um Wissen einfach automatisiert zu verarbeiten und neues Wissen abzuleiten. Wenn Ontologien Informationen zu überlappenden Konzepten enthalten, werden diese Konzepte mit sogenannten Ontologie-Alignments verbunden. Da neues Wissen hinzukommen und bestehendes Wissen überarbeitet werden kann, ändert sich der Inhalt der Ontologien im Laufe der Zeit. An diese Änderungen müssen auch die Alignments angepasst werden.

Vor diesem Hintergrund war die Zielsetzung dieser Arbeit die Betrachtung von maschinellen Verfahren zur Anpassung von Alignments an Ontologien. Dabei wurden Ansätze, die Inferenzen einsetzen, und Verfahren, die maschinellen Lernens einsetzen, betrachtet.

9.1 Ergebnisse

Die Arbeit begann mit einer Einführung in Wissensgraphen und Ontologien. Zunächst wurde dazu eine allgemeine Definition von Wissensgraphen präsentiert und eine verbreitete Umsetzung dieses Konzepts, RDF, vorgestellt. Außerdem wurde die Abfragesprache für RDF-Graphen, SPARQL, präsentiert. Anschließend wurde eine formale Definition von Ontologien gegeben und anhand einer beispielhaften Beschreibungslogik gezeigt, wie die Semantik von Ontologien abgebildet werden kann. Aufbauend auf RDF und der Definition von Ontologien wurde die Ontologie-Beschriebungssprache OWL beschrieben. Ontologie-Alignments wurden im Anschluss vorgestellt. Dabei wurde auch die Erstellung von Alignments betrachtet und besonderer Fokus auf die automatisierte Erzeugung von diesen Alignments gelegt.

Außerdem wurden einige Grundlagen zu maschinellem Lernen auf Wissensgraphen aufgeführt. Dazu wurden zuerst mehrere Ansätze zum Lernen von Vektor-Repräsentationen präsentiert. Zu den präsentierten Ansätzen gehören Ansätze aus der Sprachmodellierung, translative

Modelle und faktorisierende Modelle. Zusätzlich wurden die in der Arbeit verwendeten Klassifikationsverfahren und Metriken zur Beurteilung von solchen Klassikationsverfahren vorgestellt.

Nach diesen grundlegenden Ausführungen wurden verwandte Arbeiten aus der Literatur diskutiert, die ähnliche Probleme betrachten, wie die vorliegende Arbeit. Es existieren sowohl Ansätze, die ebenfalls Alignments an Veränderungen in Ontologien anpassen als auch entfernter verwandte Arbeiten, die Veränderungen mit maschinellem Lernen betrachten oder Alignments mit maschinellem Lernen finden. Aus dieser Diskussion von verwandten Arbeiten ergaben sich mehrere Forschungslücken. Zu den zentralen Aspekten, die in der Literatur nicht betrachtet wurden, zählen die explizite Einbeziehung von Inferenzen, die von Reasonern berechnet werden, und die Anwendung von Methoden des maschinellen Lernens auf die Anpassung von Alignments.

Aus den Forschungslücken ergab sich eine im Anschluss beschriebene Problemstellung. Nach einer abstrakten Kategorisierung von möglichen Verfahren zur Anpassung von Alignments folgte eine Darstellung der Ziele der Arbeit. Diese Ziele der Arbeit wurden in der sich daran anschließenden Verallgemeinerung der Aufgabenstellung formal definiert.

Zur Bearbeitung dieser Problemstellung wurden mehrere Ansätze entwickelt: Ein Ansatz, der Inferenzen mit Regeln zur Vorhersage von Änderungen am Alignment verbindet, ein Ansatz, der Embeddings verwendet, um die strukturellen Aspekte der Ontologien zu repräsentieren, und verbreitete Klassifikationsverfahren anwendet, ein Ansatz, der Embeddings und ein Veränderungsmodell kombiniert und ein Ansatz, der eine speziell auf Wissensgraphen ausgerichtet Netzwerkarchitektur verwendet. Alle Ansätze wurden auf Basis der in der Problemstellung vorgestellten Notation formalisiert. Für den Ansatz zur Einbeziehung von Inferenzen ergaben sich aus der Formulierung bereits einige Einschränkungen, die bei der Verwendung des Ansatzes beachtet werden müssen.

Die Umsetzung der Ansätze wurde prototypisch implementiert: Zur Anwendung des Ansatzes zur regelbasierten Einbeziehung von Inferenzen wurde ein Framework erstellt. Dieses Framework verwendet eine im Rahmen dieser Arbeit erstellte Erweiterung des Protégé-Plugins owl-diff zur Erkennung von Veränderungen. Alle Ansätze, die maschinelles Lernen verwendenden wurden mit einer ähnlichen Struktur implementiert, damit eine vergleichbare Evaluation für diese Ansätze durchgeführt werden kann.

Mithilfe der Umsetzung wurde eine Evaluation durchgeführt. Da der Ansatz zur Einbeziehung von Inferenzen und die Ansätze, die auf maschinellem Lernen aufbauen, leicht unterschiedliche Ziele verfolgen, mussten auch für die Beurteilung dieser Ansätze unterschiedliche Kriterien angewandt werden. Das führte zu einer getrennten Evaluation dieser Ansätze. Der Ansatz zur Einbeziehung von Inferenzen wurde anhand eines allgemeinen Vorgehens zum Austausch von Umgebungsdaten in sogenannten Interweaving Systems dargestellt. Dieser allgemeine Ansatz

wurde dann auf ein Beispiel aus dem Bereich der selbstorganisierten Verkehrssteuerung angewandt. Wichtigstes Ergebnis dieser Evaluation war, dass der Ansatz prinzipiell angewandt werden kann und möglicherweise den Implementierungsaufwand reduzieren kann. Außerdem ist es möglich, verschiedene Konzepte mit dieser Methode getrennt zu betrachten (separation of concerns).

Die Ansätze, die maschinelles Lernen einsetzen, wurden auf einem Datensatz durchgeführt, der aus dem biomedizinischen Metathesaurus UMLS extrahiert wurde. Es wurden zwei Experimente auf diesem Datensatz durchgeführt: einerseits wurde der Embedding-basierte Ansatz mit der Kombination aus Embeddings und Veränderungsmodell verglichen. Da hier keine Vorteile der Kombination aus Embeddings und Veränderungsmodells beobachtet werden konnten, wurde dieser Ansatz nicht weiter betrachtet. In einer Evaluation auf zwei größeren Datensätzen wurde der Ansatz zur Einbeziehung von Inferenzen mit der Anwendung von Graph Convolutional Networks verglichen. Beide Ansätze scheinen grundsätzlich auf diese Problemstellung angewendet werden zu können. Der Ansatz, der Graph Convolutional Networks einsetzt, übertraf teilweise die Leistung der Embedding-basierten Ansätze bezüglich der betrachteten Metriken.

Die Arbeit endete mit einer Diskussion der erreichten Ergebnisse. Dazu wurden zuerst die in der Problemstellung beschriebenen Forschungsfragen anhand der Ergebnisse der Evaluation beantwortet und in Beziehung zu den entwickelten Ansätzen gesetzt. Anschließend wurden die Ansätze jeweils verglichen und anhand ihrer Vor- und Nachteile bewertet. Der wichtigste Vorteil des Ansatzes zur regelbasierten Einbeziehung von Inferenzen ist, dass für die Reaktion auf Veränderungen eine sehr kompakte Formulierung möglich ist und sich deshalb beim Erstellen dieser Formulierung auf die wesentlichsten Aspekte konzentriert werden kann. Der zentrale Nachteil des Ansatzes ist, dass das Berechnen von Inferenzen auch mit den in der Arbeit vorgestellten Vereinfachungen über eine sehr hohe Laufzeitkomplexität verfügt (teilweise Klasse NEXPTIME - exponentieller Zeitaufwand in nichtdeterministischen Turingmaschinen).

Für alle Ansätze, die maschinelles Lernen einsetzen, wurde der Nachteil beschrieben, dass diese keine exakte Lösung liefern können und keine Garantie zur Güte der Ergebnisse gegeben werden kann. Für den Ansatz, der Embeddings einsetzt, wurde festgestellt, dass ein zentraler Vorteil die Möglichkeit ist, die Struktur des Graphen repräsentieren zu können, ohne das händisch Feature-Engineering angewandt werden muss. Die wichtigsten Nachteile beziehen sich auf die Laufzeit und den Speicherplatzbedarf bei der Erzeugung der Embeddings. Alleine um die Embeddings im Speicher zu halten, während sie berechnet werden, sind bis zu 10 GB Speicher notwendig. Für den Ansatz, der Embeddings mit Veränderungsmodellen kombiniert, konnten keine Vorteile gefunden werden. Als mögliche Gründe dafür, dass sich keine Vorteile ergeben wurden ein möglicherweise ungeeignetes Veränderungsmodell oder ein Klassifikationsproblem,

das zur Einbeziehung eines Veränderungsmodells nicht besonders gut geeignet ist aufgeführt.

Für die Graph Convolutional Networks wurde als zentraler Vorteil genannt, dass dieser Ansatz in der Lage ist, den Embedding-basierten Ansatz teilweise zu übertreffen. Als möglicher Grund dafür wurde die Anpassbarkeit der Knoten-Repräsentation gesehen, da das Netz diese Repräsentation für Knoten anpassen kann. Als ein zentraler Nachteil dieses Ansatzes wurde der Arbeitsspeicher-Bedarf genannt. Da für jedes Layer des Netzes eine Repräsentation jedes Knotens im Netz im Arbeitsspeicher gehalten werden muss und die komplette Struktur des Graphen ebenfalls im Netz enthalten sein muss, ist die mögliche Tiefe des Netzes für große Graphen äußerst beschränkt.

Außerdem wurden die Grenzen der in dieser Arbeit durchgeführten Untersuchungen betrachtet. Zu den genannten Einschränkungen gehört unter anderem, dass unklar ist, wie sich die Vor- und Nachteile des Ansatzes zur Einbeziehung von Inferenzen bei einer Übertragung des Ansatzes in der Praxis verhalten. Die Grenzen dieses Ansatzes konnten für eine Anwendung in der Praxis nicht vollständig aufgezeigt werden. Für alle Ansätze des maschinellen Lernens wurde als Einschränkung festgestellt, dass die Aussagen über die Leistung der Ansätze auf die in der Arbeit betrachteten Datensätze beschränkt sind. Beim Ansatz, der Embeddings und Klassifikationsverfahren kombiniert, konnten nicht alle Parameter des Verfahrens betrachtet werden, da es zu viele Kombinationsmöglichkeiten gegeben hätte.

Am Schluss der Diskussion wurden einige über den Kontext der in dieser Arbeit betrachteten Fragestellung hinaus interessanten Beobachtungen aufgeführt. Dazu wurde einerseits festgestellt, dass viele Aussagen über die in der Arbeit betrachteten Graphknoten nur anhand ihrer näheren Nachbarschaft getroffen werden können und dass sich teilweise sehr viele Informationen nur aus der Graphstruktur ziehen lassen. Dies deckt sich teilweise mit Ergebnissen, die im Bereich der Sprachmodellierung erzielt wurden. Außerdem hat sich gezeigt, dass die Komplexität der Berchnung von Inferenzen insbesondere für große Ontologien ein nur schwer beherrschbares Problem darstellt, auch wenn das Problem mit der in der Arbeit betrachteten reduzierten Inferenz etwas einfacher handhabbar ist.

9.2 Ausblick

Es bestehen zahlreiche Anknüpfungspunkte für zukünftige Arbeiten. Eine mögliche Erweiterung dieser Arbeit wäre die Anwendung des Ansatzes zur Einbeziehung von Inferenzen auf Interweaving Systems in einem realen Anwendungsfall, um die Vor- und Nachteile des Ansatzes in der Praxis beobachten zu können. Dazu würde entweder eine entsprechende Simulationsumgebung oder eine Möglichkeit benötigt, diesen Ansatz in der Praxis anwenden zu können. Durch eine

solche Erweiterung könnten auch andere Anwendungsbereiche besser beurteilt werden.

Außerdem wäre es denkbar, Verfahren zum maschinellen Lernen von Regeln auf den Ansatz zum regelbasierten Einbeziehen von Inferenzen anzuwenden. Diese Kombination von Verfahren würde möglicherweise auch eine Anwendung auf größeren Datensätzen ermöglichen. Um diese Kombination von Verfahren einsetzen zu können, muss allerdings erst noch ein Systematisches Vorgehen zum Erzeugen von Regeln für den in dieser Arbeit behandelten Anwendungsfall der Anpassung von Alignments an Ontologie-Veränderungen gefunden werden.

Für alle Verfahren, die auf dem Biomedizinischen Datensatz angewandt wurden, sind Erweiterungen auf weitere Datensätze denkbar. Hier wird es allerdings eine Herausforderung darstellen, entsprechende Datensätze zu finden, da, wie in der Arbeit bereits geschildert, einige Informationen vorliegen müssen, um den Ansatz sinnvoll evaluieren zu können. Eine solche Evaluation ist vor der Anwendung des Ansatzes auf anderen Beispielen zwingend erforderlich, da ansonsten nur schwer eingeschätzt werden kann, über welche Qualität die Vorschläge des Ansatzes verfügen. Ein möglicher Datensatz dazu wäre die Verbindung zwischen Wikidata und DBPedia. Diese enzykoplädischen Datensätze verfügen über eine Änderungshistorie und Verbindungen zwischen den in den Konzepten abgelegten Datensätzen. Es wäre also möglich die Datensätze als Ontologien mit Alignments zu betrachten. Es wäre allerdings ein sehr großer Aufwand diese Abbildung vorzunehmen und Verfahren des maschinellen Lernens darauf anzuwenden, da die Datensätze sehr groß sind.

Eine weitere interessante Weiterentwicklung dieser Arbeit wäre die Anwendung von weiteren in dieser Arbeit nicht betrachteten Embedding-Verfahren. Auch eine Anwendung von RDF2Vec auf einem großen Datensatz, für das eine große Menge an Speicher benötigt wird, könnte weitere Erkenntnisse bringen. Diese Anwendung auf größeren Daten wurde in dieser Arbeit nicht betrachtet, da dies mit den zur Verfügung stehenden Ressourcen nicht möglich war. Es wurde kürzlich mit RDF2Vec light eine Erweiterung von RDF2Vec vorgestellt, die nur Embeddings für tatsächlich betrachtete Knoten erstellt [Por+20]. Ein solches Verfahren könnte den Nachteil des Embedding-basierten Ansatzes ausgleichen, dass immer Embeddings für den gesamten Graphen erzeugt werden müssen.

Auch eine Aufbereitung des Graphen, der als Eingabe für die Embedding-Verfahren verwendet wird, könnte weiter untersucht werden. Es wäre zum Beispiel denkbar, für die Umgebung der besonders relevanten Knoten im Graph Inferenzen zu berechnen und diese dann dem Graph hinzuzufügen. Dann kann auf dem erweiterten Graphen ein Embedding trainiert werden. So könnten für die relevanten Knoten noch mehr semantische Informationen hinzugezogen werden.

Außerdem könnten für die Kombination von Embeddings und Veränderungsmodellen weitere Veränderungsmodelle untersucht werden. Da dazu erst Implementierungen von anderen

Veränderungsmodellen erstellt werden müssten, ist dafür ein erheblicher Aufwand erforderlich. Eine weitere Möglichkeit Veränderungsmodelle und auf maschinellem Lernen aufbauende Ansätze zu kombinieren wäre ein in zwei Schritte aufgeteilter Prozess: im ersten Schritt würde mithilfe der auf maschinellem Lernen basierenden Ansätze festgestellt, ob ein Veränderungsbedarf aufgrund von bestimmten Änderungen besteht, im zweiten Schritt würde je nach Art der Veränderung eine Regel angewandt, die das Alignment anpasst. So könnte möglicherweise eine höhere Präzision beim Anpassen der Embeddings erzielt werden.

Literatur

[AG08] Renzo Angles und Claudio Gutierrez. „The expressive power of SPARQL“. In: *Proceedings of the International Semantic Web Conference*. Springer. 2008, S. 114–129.

[Ali+20] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp und Jens Lehmann. *Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework*. 2020. arXiv: 2006.13365 [cs.AI].

[Aue+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak und Zachary Ives. „DBpedia: A Nucleus for a Web of Open Data“. In: *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*. ISWC'07/ASWC'07. Busan, Korea: Springer-Verlag, 2007, S. 722–735. ISBN: 3-540-76297-3, 978-3-540-76297-3.

[BL+01] Tim Berners-Lee, James Hendler und Ora Lassila. „The semantic web“. In: *Scientific american* 284.5 (2001), S. 34–43.

[BS07] Stephan Bloehdorn und York Sure. „Kernel methods for mining instance data in ontologies“. In: *The Semantic Web*. Springer, 2007, S. 58–71.

[Baa+10] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi und Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd. USA: Cambridge University Press, 2010. ISBN: 0521150116.

[Ben12] Tim Benson. *Principles of health interoperability HL7 and SNOMED*. Springer Science & Business Media, 2012.

[Bod04] Olivier Bodenreider. „The Unified Medical Language System (UMLS): integrating biomedical terminology“. In: *Nucleic Acids Research* 32 (Jan. 2004), S. D267–D270. ISSN: 0305-1048. DOI: 10.1093/nar/gkh061.

[Bor+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston und Oksana Yakhnenko. „Translating embeddings for modeling multi-relational data“. In: *Advances in neural information processing systems*. 2013, S. 2787–2795.

[Bre+84] Leo Breiman, Jerome Friedman, Charles J Stone und Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[Bro+20] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. *Language models are few-shot learners*. 2020. arXiv: 2005.14165 [cs.AI].

[Bus+19] Dan Busbridge, Dane Sherburn, Pietro Cavallo und Nils Y Hammerla. *Relational graph attention networks*. 2019. arXiv: 1904.05811 [cs.AI].

[Car+18] Silvio Domingos Cardoso, Cédric Pruski und Marcos Da Silveira. „Supporting biomedical ontology evolution by identifying outdated concepts and the required type of change“. In: *Journal of Biomedical Informatics* 87 (2018), S. 1 –11. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2018.08.013.

[Che+16] Muhao Chen, Yingtao Tian, Mohan Yang und Carlo Zaniolo. *Multilingual knowledge graph embeddings for cross-lingual knowledge alignment*. 2016. arXiv: 1611.03954 [cs.AI].

[Chi17] Davide Chicco. „Ten quick tips for machine learning in computational biology“. In: *BioData mining* 10.1 (2017), S. 35.

[Con04] Gene Ontology Consortium. „The Gene Ontology (GO) database and informatics resource“. In: *Nucleic acids research* 32 (2004).

[Cru09] Isabel F Cruz. „AgreementMaker : Efficient Matching for Large Real-World Schemas and Ontologies“. In: *PVLDB* 2.2 (2009), S. 1586–1589. ISSN: 21508097. DOI: 10.14778/1687553.1687598.

[DR+14] Julio Cesar Dos Reis, Cédric Pruski, Marcos Da Silveira und Chantal Reynaud-Delaître. „Understanding semantic mapping evolution by observing changes in biomedical ontologies“. In: *Journal of biomedical informatics* 47 (2014), S. 71–82.

[EM09] Orri Erling und Ivan Mikhailov. „RDF Support in the Virtuoso DBMS“. In: *Networked Knowledge-Networked Media*. Springer, 2009, S. 7–24.

[ES07] Jérôme Euzenat und Pavel Shvaiko. *Ontology matching*. Heidelberg: Springer, 2007, S. 333. ISBN: 3540496114. DOI: 10.1007/978-3-540-49612-0.

[FR15] Michael Färber und Achim Rettinger. „A Statistical Comparison of Current Knowledge Bases“. In: *Joint Posters and Demos Track of 11th International Conference on Semantic Systems, Posters and Demos@ SEMANTiCS 2015 and 1st Workshop on Data Science: Methods, Technology and Applications, DSci 2015-co-located with the 11th International Conference on Semantic Systems, SEMANTiCS 2015; Vienna; Austria; 15 September 2015 through 17 September 2015. Ed.: A. Polleres.* 2015, S. 18.

[Fal+14] Riccardo Falco, Aldo Gangemi, Silvio Peroni, David Shotton und Fabio Vitali. „Modelling OWL ontologies with Graffoo“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8798 (2014), S. 320–325. ISSN: 16113349. DOI: 10.1007/978-3-319-11955-7_42.

[Fer+16] Susel Fernandez, Rafik Hadfi, Takayuki Ito, Ivan Marsa-Maestre und Juan Velasco. „Ontology-Based Architecture for Intelligent Transportation Systems Using a Traffic Sensor Network“. In: *Sensors* 16.8 (2016), S. 1287. ISSN: 1424-8220. DOI: 10.3390/s16081287.

[Flo62] Robert W. Floyd. „Algorithm 97: Shortest Path“. In: *Commun. ACM* 5.6 (Juni 1962), S. 345. ISSN: 0001-0782. DOI: 10.1145/367766.368168.

[GS04] Fausto Giunchiglia und Pavel Shvaiko. „Semantic Matching“. In: *Knowledge Engineering Review Journal* 18.3 (2004), S. 265–280. ISSN: 1469-8005.

[GW97] Bernhard Ganter und Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. 1st. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. ISBN: 3540627715.

[Gam+95] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Design Patterns Elements of reusable object-oriented sofware*. Addison Wesley, 1995.

[Goo+16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618, 9780262035613.

[Gro+13] Anika Groß, Julio Cesar dos Reis, Michael Hartung, Cédric Pruski und Erhard Rahm. „Semi-automatic adaptation of mappings between life science ontologies“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7970 LNBI (2013), S. 90–104. ISSN: 03029743. DOI: 10.1007/978-3-642-39437-9_8.

[Gro+16] Anika Groß, Cédric Pruski und Erhard Rahm. „Evolution of biomedical ontologies and mappings: overview of recent approaches“. In: *Computational and structural biotechnology journal* 14 (2016), S. 333–340.

[Gru+93] Thomas R Gruber et al. „A translation approach to portable ontology specifications“. In: *Knowledge acquisition* 5.2 (1993), S. 199–220.

[Han+18] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun und Juanzi Li. „OpenKE: An Open Toolkit for Knowledge Embedding“. In: *Proceedings of EMNLP*. 2018.

[Har+13] Michael Hartung, Anika Groß und Erhard Rahm. „COnto-Diff: Generation of complex evolution mappings for life science ontologies“. In: *Journal of Biomedical Informatics* 46.1 (2013), S. 15–32. ISSN: 15320464. DOI: 10.1016/j.jbi.2012.04.009.

[Har+19] Ian Harrow, Rama Balakrishnan, Ernesto Jimenez-Ruiz, Simon Jupp, Jane Lomax, Jane Reed, Martin Romacker, Christian Senger, Andrea Splendiani, Jabe Wilson et al. „Ontology mapping for semantically enabled applications“. In: *Drug discovery today* (2019).

[Hep08] Martin Hepp. „Goodrelations: An ontology for describing products and services offers on the web“. In: *International conference on knowledge engineering and knowledge management*. Springer. 2008, S. 329–346.

[Hit+09] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider und Sebastian Rudolph, Hrsg. *OWL 2 Web Ontology Language: Primer*. Verfügbar unter http://www.w3.org/TR/owl2-primer/, [zuletzt abgerufen: Dezember 2020]. W3C Recommendation, 2009.

[Ho95] Tin Kam Ho. „Random decision forests“. In: *Proceedings of 3rd international conference on document analysis and recognition*. Bd. 1. IEEE. 1995, S. 278–282.

[Hog+20] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab und Antoine Zimmermann. *Knowledge Graphs*. 2020. arXiv: 2003.02320 [cs.AI].

[Hol18] Andreas Holzinger. „Explainable AI (ex-AI)“. In: *Informatik-Spektrum* 41.2 (2018), S. 138–143.

[Hor+06] Ian Horrocks, Oliver Kutz und Ulrike Sattler. „The Even More Irresistible SROIQ.“ In: *Knowledge Representation* 6 (2006), S. 57–67.

[Hu+08] Wei Hu, Yuzhong Qu und Gong Cheng. „Matching large ontologies: A divide-and-conquer approach“. In: *Data & Knowledge Engineering* 67.1 (Okt. 2008), S. 140–160. ISSN: 0169023X. DOI: 10.1016/j.datak.2008.06.003.

[JI17] Matthias Jurisch und Bodo Igler. „Knowledge-Based Self-Organization of Traffic Control Systems“. In: *47. Jahrestagung der Gesellschaft für Informatik, Informatik 2017, Chemnitz, Germany, September 25-29, 2017*. 2017, S. 947–954. DOI: 10.18420/in2017_97.

[JI18] Matthias Jurisch und Bodo Igler. „RDF2Vec-based Classification of Ontology Alignment Changes“. In: *Proceedings of the First Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conerence (ESWC 2018)*. 2018.

[JI19] Matthias Jurisch und Bodo Igler. „Graph-Convolution-Based Classification for Ontology Alignment Change Prediction“. In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.* 2019, S. 11–20. URL: http://ceur-ws.org/Vol-2377/paper_2.pdf.

[JI20] Matthias Jurisch und Bodo Igler. „Modelling of Change Response in Interweaving Systems as Ontology Alignment Adaption“. In: *50. Jahrestagung der Gesellschaft für Informatik, Informatik 2020, Germany, September 28-October 2, 2020*. 2020, S. 947–954.

[JM+09] Yves R. Jean-Mary, E. Patrick Shironoshita und Mansur R. Kabuka. „Ontology matching with semantic verification“. In: *Journal of Web Semantics* 7.3 (2009), S. 235–251. ISSN: 15708268. DOI: 10.1016/j.websem.2009.04.001.

[JR+20] Ernesto Jiménez-Ruiz, Asan Agibetov, Jiaoyan Chen, Matthias Samwald und Valerie Cross. „Dividing the Ontology Alignment Task with Semantic Embeddings and Logic-based Modules“. In: *arXiv preprint arXiv:2003.05370* (2020).

[Jac05] J Edward Jackson. *A user's guide to principal components*. Bd. 587. John Wiley & Sons, 2005.

[Ji+15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu und Jun Zhao. „Knowledge graph embedding via dynamic mapping matrix“. In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, S. 687–696.

[Jr+10] Ernesto Jiménez-ruiz, Bernardo Cuenca Grau, Ian Horrocks und Rafael Berlanga. „Logic-based assessment of the compatibility of UMLS ontology sources“. In: *Journal of Biomedical Semantics*. 2010. DOI: 10.1186/2041-1480-2-S1-S2.

[Jr20] Ernesto Jiménez-ruiz. *Ontology Alignment Evaluation Initiative (OAEI)*. Webseite, http://oaei.ontologymatching.org/ [Zuletzt abgerufen: August 2020]. 2020.

[Jur16] Matthias Jurisch. „Managing Ontology Mapping Change based on Changing Inference Sets“. In: *Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-a-LOD. Bologna, Italy, November 19-23, 2016. Revised Selected Papers*. Springer, 2016, 255–262.

[KP18] Seyed Mehran Kazemi und David Poole. „Simple embedding for link prediction in knowledge graphs“. In: *Advances in neural information processing systems*. 2018, S. 4284–4295.

[KP20] Adrianna Kozierkiewicz. und Marcin Pietranik. „Updating Ontology Alignment on the Relation Level based on Ontology Evolution“. In: *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,* INSTICC. SciTePress, 2020, S. 241–248. ISBN: 978-989-758-421-3. DOI: 10.5220/0009142002410248.

[KW17] Thomas N. Kipf und Max Welling. „Semi-Supervised Classification with Graph Convolutional Networks“. In: *Proc. of the International Conference on Learning Representations (ICLR)*. 2017.

[Kad+17] Rudolf Kadlec, Ondrej Bajgar und Jan Kleindienst. „Knowledge Base Completion: Baselines Strike Back“. In: *ACL 2017* (2017), S. 69.

[LT06] Patrick Lambrix und He Tan. „SAMBO—A system for aligning and merging biomedical ontologies“. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 4.3 (Sep. 2006), S. 196–206. ISSN: 15708268. DOI: 10.1016/j.websem.2006.05.003. URL: http://www.sciencedirect.com/science/article/pii/S1570826806000151.

[Li+09] Juanzi Li, Jie Tang, Yi Li und Qiong Luo. „RiMOM: A dynamic multistrategy ontology alignment framework“. In: *IEEE Transactions on Knowledge and Data Engineering* 21.8 (2009), S. 1218–1232. ISSN: 10414347. DOI: 10.1109/TKDE.2008.202.

[Lin+15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Y Liu und X Zhu. „Learning Entity and Relation Embeddings for Knowledge Graph Completion“. In: *AAAI 2015*. 2015.

[Lin+93] Donald AB Lindberg, Betsy L Humphreys und Alexa T McCray. „The unified medical language system“. In: *Methods of information in medicine* 32.4 (1993), S. 281.

[MVH+04] Deborah L McGuinness, Frank Van Harmelen et al. „OWL web ontology language overview“. In: *W3C recommendation* 10.10 (2004), S. 2004.

[Mei+18] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla und Heiner Stuckenschmidt. „Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion“. In: *International Semantic Web Conference*. Springer. 2018, S. 3–20.

[Mik+13] Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean. *Efficient estimation of word representations in vector space*. 2013. arXiv: 1301.3781 [cs.AI].

[Mot+12] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue und Carsten Lutz. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C Standard, https://www.w3.org/TR/owl2-profiles/#Computational_Properties, [zuletzt abgerufen: Dezember 2020]. 2012.

[Mot07] Boris Motik. „On the properties of metamodeling in OWL“. In: *Journal of Logic and Computation* 17.4 (2007), S. 617–637.

[NM04] Natalya F Noy und Mark a Musen. „Ontology Versioning in an Ontology Management Framework“. In: *IEEE Intelligent Systems* (2004), S. 6–13. ISSN: 15411672. DOI: 10.1109/MIS.2004.33.

[NVV11] M Nagy und M Vargas-Vera. „Multiagent Ontology Mapping Framework for the Semantic Web“. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 41.4 (Juli 2011), S. 693–704. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2011.2132704.

[Nel+01] Stuart J Nelson, W Douglas Johnston und Betsy L Humphreys. „Relationships in medical subject headings (MeSH)“. In: *Relationships in the Organization of Knowledge*. Springer, 2001, S. 171–184.

[Nic+11] Maximilian Nickel, Volker Tresp und Hans-Peter Kriegel. „A three-way model for collective learning on multi-relational data.“ In: *Proceedings of the 28th International Conference on Machine Learning*. Bd. 11. 2011, S. 809–816.

[Noy+01] Natalya F Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W Fergerson und Mark A Musen. „Creating semantic web contents with protege-2000“. In: *IEEE intelligent systems* 16.2 (2001), S. 60–71.

[OC+15] Lorena Otero-Cerdeira, Francisco J. Rodríguez-Martínez und Alma Gómez-Rodríguez. „Ontology Matching: A Literature Review“. In: *Expert Systems With Applications* 42.2 (2015), S. 949–971. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2014.08.032.

[Ode+08] Martin Odersky, Lex Spoon und Bill Venners. *Programming in scala*. Artima Inc, 2008.

[PC12] Catia Pesquita und Francisco M. Couto. „Predicting the Extension of Biomedical Ontologies“. In: *PLOS Computational Biology* 8.9 (Sep. 2012), S. 1–16. DOI: 10.1371/journal.pcbi.1002630.

[Pau18] Heiko Paulheim. „How much is a Triple ? Estimating the Cost of Knowledge Graph Creation“. In: *Proc. of the International Semantic Web Conference 2018* (2018), S. 5–8.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[Pen+14] Jeffrey Pennington, Richard Socher und Christopher D. Manning. „GloVe: Global Vectors for Word Representation“. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, S. 1532–1543.

[Por+20] Jan Portisch, Michael Hladik und Heiko Paulheim. *RDF2Vec Light – A Lightweight Approach for Knowledge Graph Embeddings*. 2020. arXiv: 2009.07659 [cs.AI].

[Pro+09] Holger Prothmann, Jürgen Branke, Hartmut Schmeck, Sven Tomforde, Fabian Rochner, J. Hahner und C. Muller-Schloer. „Organic Traffic Light Control for Urban Road Networks“. In: *International Journal of Autonomous and Adaptive Communications Systems* 2.3 (2009), S. 203–225. ISSN: 1754-8632. DOI: 10.1504/IJAACS.2009.026783.

[RMJ03] Cornelius Rosse und José L. V. Mejino Jr. „A reference ontology for biomedical informatics: the Foundational Model of Anatomy“. In: *Journal of biomedical informatics* 36.6 (2003), S. 478–500.

[RN10] Stuart Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3. Aufl. Prentice Hall, 2010.

[RN11] Timothy Redmond und Natasha Noy. „Computing the changes between ontologies“. In: *Joint Workshop on Knowledge Evolution and Ontology Dynamics*. 2011, S. 1–14.

[RP16] Petar Ristoski und Heiko Paulheim. „RDF2Vec: RDF Graph Embeddings for Data Mining“. In: *The Semantic Web - ISWC 20162016*. 2016, S. 498–514. DOI: 10.1007/978-3-319-46523-4_30.

[Rei+15a] Julio Cesar dos Reis, Cédric Pruski, Marcos Da Silveira und Chantal Reynaud-Delaître. „The DyKOSMap Approach for Analyzing and Supporting the Mapping Maintenance Problem in Biomedical Knowledge Organization Systems“. In: *The Semantic Web: ESWC 2012 Satellite Events*. Hrsg. von Elena Simperl, Barry Norton, Dunja Mladenic, Emanuele Della Valle, Irini Fundulaki, Alexandre Passant und Raphaël Troncy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, S. 163–175. ISBN: 978-3-662-46641-4. DOI: 10.1007/978-3-662-46641-4_12.

[Rei+15b] Julio dos Reis, Cédric Pruski, Marcos Silveira und Chantal Reynaud. „DyKOSMap: A Framework for Mapping Adaptation between Biomedical Knowledge Organization Systems“. In: *Journal of biomedical informatics* 55 (Apr. 2015). DOI: 10.1016/j.jbi.2015.04.001.

[SA09] Md. Hanif Seddiqui und Masaki Aono. „An Efficient and Scalable Algorithm for Segmented Alignment of Ontologies of Arbitrary Size“. In: *Web Semant.* 7.4 (Dez. 2009), S. 344–356. ISSN: 1570-8268. DOI: 10.1016/j.websem.2009.09.001.

[SE13] Pavel Shvaiko und Jerome Euzenat. „Ontology Matching: State of the Art and Future Challenges“. In: *IEEE Transactions on Knowledge and Data Engineering* 25.X (2013), S. 158–176. ISSN: 10414347. DOI: 10.1109/TKDE.2011.253.

[Sch+18] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov und Max Welling. „Modeling Relational Data with Graph Convolutional Networks“. In: *The Semantic Web*. Hrsg. von Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai und Mehwish Alam. Cham: Springer International Publishing, 2018, S. 593–607. ISBN: 978-3-319-93417-4. DOI: 10.1007/978-3-319-93417-4_38.

[Shi+19] Chaitrali Dipak Shirke, Ashish Bhaskar und Edward Chung. „Australia and New Zealand“. In: *Global practices on road traffic signal control: Fixed-time control at isolated intersections (World Conferences on Transport Research):* Hrsg. von K Tang, Z Tian, M Boltze und H Nakamura. Netherlands: Elsevier Inc., 2019, S. 139–162. DOI: 10.1016/B978-0-12-815302-4.00009-1.

[Sin12] Amit Singhal. *Introducing the Knowledge Graph: things, not strings.* https://www.blog.google/products/search/introducing-knowledge-graph-things-not/, zuletzt abgerufen: 1. Mai 2020. 2012.

[Sio+07] Nicholas Sioutos, Sherri de Coronado, Margaret W. Haber, Frank W. Hartel, Wen-Ling Shaiu und Lawrence W. Wright. „NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information“. In: *Journal of Biomedical Informatics* 40.1 (2007). Bio*Medical Informatics, S. 30 –43. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2006.02.013.

[Sir+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur und Yarden Katz. „Pellet: A practical owl-dl reasoner“. In: *Journal of Web Semantics* 5.2 (2007), S. 51–53.

[Ste+16] Anthony Stein, Sven Tomforde und Dominik Rauh. „Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gap“. In: *IEEE International Conference on Autonomic Computing*. 2016. DOI: 10.1109/ICAC.2016.20.

[Sto+02] L. Stojanovic, Alexander Maedche, Boris Motik und Nenad Stojanovic. „User-driven ontology evolution management“. In: *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web* (2002), S. 133–140.

[Stu+98] Rudi Studer, V Richard Benjamins und Dieter Fensel. „Knowledge engineering: principles and methods“. In: *Data & knowledge engineering* 25.1-2 (1998), S. 161–197.

[Stu10] H. Stuckenschmidt. *Ontologien.* Informatik. Berlin und Heidelberg: Springer, 2010. ISBN: 9783642114595.

[Sun+17] Zequn Sun, Wei Hu und Chengkai Li. „Cross-lingual entity alignment via joint attribute-preserving embedding“. In: *Proceedings of the International Semantic Web Conference*. Springer. 2017, S. 628–644.

[Sun+18] Zequn Sun, Wei Hu, Qingheng Zhang und Yuzhong Qu. „Bootstrapping Entity Alignment with Knowledge Graph Embedding.“ In: *IJCAI*. Bd. 18. 2018, S. 4396–4402.

[Sun+19] Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang und Yuzhong Qu. *Knowledge Graph Alignment Network with Gated Multi-hop Neighborhood Aggregation*. 2019. arXiv: 1911.08936 [cs.CL].

[Sut+14] Ilya Sutskever, Oriol Vinyals und Quoc V. Le. „Sequence to Sequence Learning with Neural Networks“. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, S. 3104–3112.

[TH06] Dmitry Tsarkov und Ian Horrocks. „FaCT++ description logic reasoner: System description“. In: *International joint conference on automated reasoning*. Springer. 2006, S. 292–297.

[Thi+19] Elodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez und Cassia Trojahn. „Survey on complex ontology matching“. In: *Semantic Web* (2019), S. 1–39.

[Tom+14] Sven Tomforde, Jörg Hähner und Bernhard Sick. „Interwoven Systems“. In: *Informatik-Spektrum* 37.5 (2014), S. 483–487. ISSN: 0170-6012. DOI: 10.1007/s00287-014-0827-z.

[Tom+16] S. Tomforde, S. Rudolph, K. Bellman und R. Würtz. „An Organic Computing Perspective on Self-Improving System Interweaving at Runtime“. In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. 2016, S. 276–284. DOI: 10.1109/ICAC.2016.15.

[Tro+16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier und Guillaume Bouchard. „Complex embeddings for simple link prediction“. In: *International Conference on Machine Learning*. 2016, S. 2071–2080.

[Tsa+13] George Tsatsaronis, Iraklis Varlamis, Nattiya Kanhabua und Kjetil Nørvåg. „Temporal classifiers for predicting the expansion of medical subject headings“. In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2013, S. 98–113.

[VE97] Petko Valtchev und Jerome Euzenat. „Dissimilarity Measure for Collections of Objects and Values“. In: *Liu X., Cohen P., Berthold M. (eds) Advances in Intelligent Data Analysis Reasoning about Data, Lecture Notes in Computer Science*. Springer, 1997, S. 259–272.

[VK14] Denny Vrandečić und Markus Krötzsch. „Wikidata: A Free Collaborative Knowledgebase“. In: *Commun. ACM* 57.10 (Sep. 2014), S. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489.

[Vel+03] Yannis Velegrakis, Renée J. Miller und Lucian Popa. „Mapping adaptation under evolving schemas“. In: *VLDB '03 Proceedings of the 29th international conference on Very Large Data Bases - Volume 29* (2003), S. 584–595.

[Wan+14] Zhen Wang, Jianwen Zhang, Jianlin Feng und Zheng Chen. „Knowledge graph embedding by translating on hyperplanes“. In: *Twenty-Eighth AAAI conference on artificial intelligence*. 2014.

[War62] Stephen Warshall. „A Theorem on Boolean Matrices“. In: *J. ACM* 9.1 (Jan. 1962), 11–12. ISSN: 0004-5411. DOI: 10.1145/321105.321107.

[Wik20a] Wikidata. *Wikidata Query Examples*. https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries/examples, Webseite [Zuletzt abgerufen: August 2020]. 2020.

[Wik20b] Wikidata. *Wikidata Statistics*. https://www.wikidata.org/wiki/Wikidata:Statistics, Webseite [Zuletzt abgerufen: August 2020]. 2020.

[Woo+14] David Wood, Markus Lanthaler und Richard Cyganiak. *RDF 1.1 Concepts and Abstract Syntax*. [zuletzt abgerufen: Dezember 2020]. Feb. 2014. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[Wu+20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang und S Yu Philip. „A comprehensive survey on graph neural networks“. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[YP05] Cong Yu und Lucian Popa. „Semantic Adaptation of Schema Mappings when Schemas Evolve“. In: *Very Large Data Bases* (2005), S. 1006 –1017.

[Yan+14] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao und Li Deng. *Embedding entities and relations for learning and inference in knowledge bases*. 2014. arXiv: 1412.6575 [cs.AI].

[ZW15] Ye Zhang und Byron Wallace. *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*. 2015. arXiv: 1510.03820 [cs.AI].

[Zha+17] Denghui Zhang, Manling Li, Yantao Jia, Yuanzhuo Wang und Xueqi Cheng. „Efficient parallel translating embedding for knowledge graphs“. In: *Proceedings of the International Conference on Web Intelligence*. 2017, S. 460–468.

[Zhu+17] Hao Zhu, Ruobing Xie, Zhiyuan Liu und Maosong Sun. „Iterative Entity Alignment via Joint Knowledge Embeddings.“ In: *IJCAI*. Bd. 17. 2017, S. 4258–4264.

[Zhu04] Mu Zhu. „Recall, precision and average precision“. In: *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* 2 (2004), S. 30.

[Apa20] Apache Foundation. *Apache Jena*. https://jena.apache.org/, Webseite [zuletzt abgerufen: August 2020]. 2020.

[For12] Forschungsgesellschaft für Straßen- und Verkehrswesen. *Begriffsbestimmungen*. Köln: FGSV Verlag, 2012.

[Git20] Github. *Openllet*. Github Repository, https://github.com/Galigator/openllet, [zuletzt Abgerufen: September 2020]. 2020.

[Goo08] Google. *gson*. Github Repository, https://github.com/google/gson, [zuletzt abgerufen: September 2020]. 2008.

[Kre+11] P. Kremen, M. Smid und Z. Kouba. „OWLDiff: A Practical Tool for Comparison and Merge of OWL Ontologies“. In: *2011 22nd International Workshop on Database and Expert Systems Applications*. 2011, S. 229–233. DOI: 10.1109/DEXA.2011.62.

[Par03] G. Pardo-Castellote. „OMG Data-Distribution Service: architectural overview“. In: *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* 2003, S. 200–206. DOI: 10.1109/ICDCSW.2003.1203555.

[Pyh07] Pyhton Software Foundation. *FMC Quick Introduction*. Webseite, http://www.fmc-modeling.org/quick-intro, zuletzt Abgerufen: Dezember 2020. 2007.

[Pyt20] Python Software Foundation. *pickle – Python Object Serialization*. Python API Dokumentation, https://docs.python.org/3/library/pickle.html, zuletzt Abgerufen: September 2020. 2020.

[Sta20] Stardog. *Clark and Pasia*. https://stardog.com/ [zuletzt abgerufen: August 2020]. 2020.

[dos+15] Julio Cesar dos Reis, Cédric Pruski, Marcos Da Silveira und Chantal Reynaud-Delaître. „DyKOSMap: A framework for mapping adaptation between biomedical knowledge organization systems“. In: *Journal of Biomedical Informatics* 55 (2015), S. 153 –173. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2015.04.001.

[rdf20] rdflib Devlopers. *rdflib 5.0.0*. https://rdflib.readthedocs.io/en/stable/, Webseite [zuletzt abgerufen: August 2020]. 2020.

Anhang A

Veröffentlichungen

Im Zusammenhang mit dieser Dissertation wurden mehrere Arbeiten publiziert. Es folgt eine kurze Auflistung dieser Veröffentlichungen.

[JI17] Matthias Jurisch und Bodo Igler. „Knowledge-Based Self-Organization of Traffic Control Systems“. In: *47. Jahrestagung der Gesellschaft für Informatik, Informatik 2017, Chemnitz, Germany, September 25-29, 2017*. 2017, S. 947–954. DOI: 10.18420/in2017_97. URL: https://doi.org/10.18420/in2017_97.

[JI18] Matthias Jurisch und Bodo Igler. „RDF2Vec-based Classification of Ontology Alignment Changes“. In: *Proceedings of the First Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conerence (ESWC 2018)*. 2018.

[JI19] Matthias Jurisch und Bodo Igler. „Graph-Convolution-Based Classification for Ontology Alignment Change Prediction“. In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.* 2019, S. 11–20. URL: http://ceur-ws.org/Vol-2377/paper_2.pdf.

[JI20] Matthias Jurisch und Bodo Igler. „Modelling of Change Response in Interweaving Systems as Ontology Alignment Adaption“. In: *50. Jahrestagung der Gesellschaft für Informatik, Informatik 2020, Germany, September 28-October 2, 2020*. 2020, S. 947–954.

[Jur16] Matthias Jurisch. „Managing Ontology Mapping Change based on Changing Inference Sets“. In: *Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-a-LOD. Bologna, Italy, November 19-23, 2016. Revised Selected Papers*. Springer, 2016, 255–262.